UNITED STATES PATENT AND TRADEMARK OFFICE

BOARD OF PATENT APPEALS AND INTERFERENCES

Applicant: White

Serial No.: 09/682,985

Group Art Unit: 2135

Confirmation No.: 4151

Attorney docket number: GREN.P001-2

For: Exchange Method and Apparatus

APPEAL BRIEF

Responsive to the Official Action mailed on August 21, 2007 for the above-captioned application (hereinafter, the Office Action), the present appeal brief is respectfully submitted.

(i) Real Party In Interest

The real party in interest is Grenex Corporation, the assignee.

(ii) Related Appeals And Interferences

There are no other appeals or interferences known to Appellant, Appellant's legal representative or Appellant's assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(iii) Status of Claims

The status of all the claims in the proceeding is as follows:

Claims	Status
1	Canceled
2	Cancelled
3	Rejected, Being Appealed
4	Cancelled
5	Cancelled

Claims	Status
6	Cancelled
7	Cancelled
8	Cancelled
9	Cancelled
10	Rejected, Being Appealed
11	Rejected, Being Appealed

The rejections of claims 3, 10 and 11 are appealed.

(iv) Status Of Amendments

No amendment has been filed subsequent to the Office Action.

(v) Summary Of Claimed Subject Matter

It is noted that this application was filed electronically, and not on paper. The electronic patent application as filed, and as contained in appellant's official electronic file, did not have and does not have page breaks or page numbers and instead only had and has paragraph numbers. Thus it is impossible to point to particular page numbers or line numbers on pages in any unambiguous way in the application as originally filed electronically.

Notwithstanding this fact about the application as filed, it is anticipated that the USPTO will take the view, even though mistaken, that this Brief is supposedly non-compliant if the appellant cites only to paragraph numbers and not to page and line numbers.

After receiving the electronic patent application, USPTO personnel hand-printed it on paper, and inserted the printed paper into the workflow of the USPTO as if the application had been filed on paper (which it was not). USPTO's hand-printing imposed artificial page breaks bearing no relation to the document in applicant's files. The artificial page breaks may be seen in the IFW (image file wrapper) system, yielding forty-six pages.

Thus to avoid a wrongful view of the brief supposedly being non-compliant, the appellant will cite not only to the paragraph numbers in the patent application as originally filed, but will also cite to the page numbers which were artificially imposed by USPTO personnel after the application was filed (and to the "line numbers" based upon the page breaks artificially imposed by USPTO personnel after filing).

It is emphasized that the citations below to page and line numbers in no way constitute an admission by the appellant that the application was filed on paper (which it was not) and in no way constitute an admission by the appellant that the application as filed had page numbers at all (which it did not).

The invention relates generally to using tcp/ip port 80 (the default port for HTTP communication) for HTTPS communication, thereby enabling secure HTTPS communication through firewalls that permit access to port 80, but block access to port 443 (the default port for HTTPS communication). See the Specification at paragraphs 230 to 246, page 30, line 6 to page 32, line 9.

NOTE: In this Application as originally filed and as currently amended, Appellant NEITHER claims the broad concept of changing the port number used for HTTPS communication NOR claims the broad concept of using HTTPS to perform secure communication. Rather, in this Application as originally filed and as currently amended, Appellant merely claims the relatively narrowly-claimed improvement of using port 80 in particular (which is normally associated with HTTP communication) for HTTPS communication.

In a **first** embodiment (i.e., claim 3) the invention includes:

(a) configuring a server program so that it listens on port 80 for requests for secure hypertext transfer protocol sessions (rather than on port 443); (see the Specification at Paragraphs 234 to 240, page 30, line 24 to page 31, line 21); in a

preferred embodiment, Internet Information Server is configured to use port 80 for HTTPS connections;

(b) receiving at the server program on port 80 a first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request is received on port 80 rather than port 443; (see the Specification at Paragraph 241, page 31, lines 22-24, and Claims 1 and 3); in a preferred embodiment, a browser requests a URL of the form "https://<address>:80" and that request is received by the server; and

(c) outputting from the server program a response to the first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request was received on port 80 rather than port 443; (see the Specification at Paragraph 241, page 31, lines 22-24, and Claims 1 and 3);

In a **second** embodiment (i.e., claim 10) the invention includes:

(a) configuring a web server system to use port 80 for communications using a protocol selected from the group consisting of: secure socket layer, secure sockets layer, SSL, secure hypertext transfer protocol, and HTTPS; (see the Specification at Paragraphs 234 to 240, page 30, line 24 to page 31, line 21);

(b) receiving at port 80 at the web server system a first data packet that is formatted in accordance with the protocol; (see the Specification at Paragraph 241, page 31, lines 22-24, and claim 10); and

(c) responding to the first data packet with a second data packet that is formatted in accordance with the protocol. (See the Specification at Paragraph 241, page 31, lines 22-24, and claim 10.)

In a **third** embodiment (i.e., claim 11) the invention includes a web server system comprising:

(a) web server software configured to use port 80 for communications using a protocol selected from the group consisting of: secure socket layer, secure sockets layer, SSL, secure hypertext transfer protocol, and HTTPS; (see the Specification at Paragraphs 234 to 240, page 30, line 24 to page 31, line 21);

(b) means for receiving at port 80 at the web server system a first data packet that is formatted in accordance with the protocol (see the Specification at Paragraphs 234 to 240, page 30, line 24 to page 31, line 21- Internet Information Server configured as described therein is such a means); and

(c) means for responding to the first data packet with a second data packet that is formatted in accordance with the protocol; (see the Specification at Paragraphs 234 to 240, page 30, line 24 to page 31, line 21 - Internet Information Server configured as described therein is such a means).

(vi) Grounds of rejection to be reviewed on appeal.

The **first** ground of rejection to be reviewed on appeal is the rejection of Claim 3 on the Examiner's view that as of the effective filing date, one skilled in the relevant art would supposedly have combined Scanlan (US Patent No. 6,029,245) and Ogdon et al (US Patent No. 6,161,137), and that the claim would supposedly have been obvious over this two-way combination.

The **second** ground of rejection to be reviewed on appeal is the rejection of Claim 10 on the Examiner's view that as of the effective filing date, one skilled in the relevant art would supposedly have combined Scanlan (US Patent No. 6,029,245) and Ogdon et al (US Patent No. 6,161,137), and that the claim would supposedly have been obvious over this two-way combination.

The **third** ground of rejection to be reviewed on appeal is the rejection of Claim 11 on the Examiner's view that as of the effective filing date, one skilled in the relevant art would supposedly have combined Scanlan (US Patent No. 6,029,245) and Ogdon et al (US Patent No. 6,161,137), and that the claim would supposedly have been obvious over this two-way combination.

(vii) Argument

Argument - Claim 3

3. Claim 3 IS patentable over Scanlan in view of Ogdon because:

3.1. None of Scanlan, Ogdon, nor Scanlan combined with Ogdon, teaches or even suggests: configuring a server program so that it listens for requests for secure hypertext transfer protocol sessions on port 80 rather than port 443.

3.1.1. Scanlan alone does NOT teach or even suggest configuring a server program so that it listens for requests for secure hypertext transfer protocol sessions on port 80 rather than port 443. In the Office action at page 2 line 22 - page 3 line 4, the Examiner (i) asserts that "Scanlan Teaches configuring the server program/system to use port 80 (rather than the first port number 443) for communications using a protocol" and (ii) in support of this assertion (and multiple other assertions discussed below) cites Scanlan at Fig.1, Fig. 2, col.1 lines 46-60, col. 3 lines 1-6, and Fig. 4 (without stating which material supports which assertions). Taking these citations one at a time:

Scanlan at Fig. 1. In Fig. 1, titled "Web Security Injection Flow", all of the arrows between Scanlan's Web Server 12 and Scanlan's Browsers 10-1, 10-2, 10-3 and 10-4 point away from the server and towards the Browsers. Consequently, nothing in Scanlan's Fig. 1 teaches configuring a server to listen for requests of any sort on any port.

Scanlan at Fig. 2. Fig. 2 contains nothing about listening, requests, protocols or ports.

Scanlan at col. 1 lines 46-60. These lines in Scanlan do at least contain the words "served" (which has the same root as "server"), "Secure Hypertext Transfer Protocol", "SHTTP", "Secure Sockets Layer", and "SSL". However, these lines do not mention port numbers at all.

Scanlan at col. 3 lines 1-6. These lines in Scanlan do at lest (i) contain the words "port" and "80" and (ii) mention that ports other than 80 can be used for HTTP (i.e., unsecured) communications. However, these lines do NOT teach or suggest configuring a server program to listen for requests for secure hypertext protocol sessions on any port, much less port 80 in particular.

Scanlan at Fig. 4. Fig. 4 contains nothing about listening, requests or ports.

Scanlan Conclusion. Thus, for the reasons discussed above, Scanlan alone neither teaches nor suggests configuring a server program so that it listens for requests for secure hypertext transfer protocol sessions on port 80 rather than port 443, despite the Examiner's assertion to the contrary.

3.1.2. Ogdon alone does NOT teach or even suggest configuring a server program so that it listens for requests for secure hypertext transfer protocol sessions on port 80 rather than port 443. In the Office action at page 3 lines 7-9, the Examiner (i) asserts that "Ogdon teaches the port 80 to communicate with the server using the various protocols selected from the group consisting of: secure socket layer, secure sockets layer, SSL, secure hypertext transfer protocol, and HTTPS" and (ii) in support of this assertion cites Ogdon at col. 16 lines 14-31.

The cited lines in Ogdon do at least contain several relevant words (e.g., "security", "port numbers", "80", "servers", "HTTPS", "Secure Sockets Protocol"), but in those lines they are not connected together in a way that is relevant to this Application. To best understand what Ogdon teaches in the lines cited by the examiner, one should find the box labeled "Security Sub-System 208" in the middle of Fig. 1.A and read straight through Ogdon's discussion of security from col. 14 line 27 to col. 18 line 19, rather than peeking through the keyhole suggested by the Examiner.

Ogdon's discussion of the Security Sub-System begins at Col. 14 lines 37 to 40 (not cited by the examiner) with the following:

> "Each host 200 is also in communication with the security subsystem 208 referred to hereinabove. Note that all external communications from third parties to a host 200 is [sic.] routed through the security subsystem 208."

At col. 14 lines 40 to 63, Ogdon describes various measures that might be employed by the Security Sub-System, including: packet filtering routers, firewalls, encryption tunnels, a validation subsystem and virus detection software.

At col. 14 line 64 to col. 15 line 2, Ogdon notes that security subsystem 208 resides on a separate computational host and may translate packets received from the network (e.g., Internet) to proprietary formats before forwarding them to other elements of Ogdon's system.

At col. 15 lines 2 to 15, Ogdon discusses how security can be weak for some presentations and strong for other presentations.

At col. 15 lines 16 to 27, Ogdon notes that the distributed content server features of Ogdon's system permit proprietary corporate data to reside behind the proprietary corporate firewall where it is unavailable even to Ogdon's system operators.

At col. 15 lines 28 to 66, Ogdon describes some of the ways in which security can vary from minimal to high.

At col. 15 line 66 to col. 16 line 3 (not cited by the Examiner), Ogdon observes that:

> "Furthermore, the high security measures may only allow network 70 connections from pre-approved network addresses using a specified protocol and port number for the duration of a particular presentation for which the client is registered."

The description of Ogdon's Security Subsystem summarized above, and the sentence quoted immediately above, make it clear that Ogdon teaches using non-standard port numbers as a means to achieve "security through obscurity". I.e., if someone with low personal moral standards who is not authorized to receive a highly secured presentation tries to view it any way by requesting information through standard port numbers, such would-be miscreant will be frustrated because the information will actually be available only through non-standard port numbers and then only if he is using a computer with a pre-approved IP address.

Ogdon at col. 16 lines 4-7 mentions that presentation data can be encrypted when it is sent over the Internet.

Ogdon at col. 16 lines 7-14 seems to be saying that if certain assumptions are satisfied, then a "presentation can implement standard web data security by using Internet protocols such as file transfer protocol (FTP) with user identification plus password, and hypertext transport protocol secure (HTTP)"

Turning now to the lines in Ogdon cited by the Examiner, and taking them one sentence at a time:

Ogdon at col. 16 lines 14-19 says:

> "Also note that the security measures for the present invention are not restricted to providing communications on generally used port numbers (e.g., communication between the host and leaders or audience members can occur on either port 60 or port 80 in any combination for a single presentation performance[])]."

Port 60 is normally unassigned. Thus, the above quoted words seem simply to reiterate the idea that using non-standard (e.g., normally unassigned) port numbers for web servers might enhance security. Nothing in the above quoted words teaches or even suggests configuring a server program so that it listens for requests for secure hypertext transfer protocol sessions on port 80 rather than port 443.

Ogdon at col. 16 lines 19 to 23 says:

> "Note that special security presentation performances can be run using any port number desired when using servers 248 in the operations center, or on intranets (e.g., the secure corporate intranet 260)."

The above quoted sentence simply reiterates the idea that using non-standard port numbers for web servers might enhance security. Nothing in the above quoted words teaches or even suggests configuring a server program so that it listens for requests for secure hypertext transfer protocol sessions on port 80 (normally assigned to HTTP sessions) rather than port 443 (normally assigned to HTTPS sessions).

The next two sentences cited by the Examiner, Ogdon at col. 16 lines 23 to 32, discuss how the HTTPS protocol can be useful for transmitting questions to participants, collecting responses from participants and returning results to participants. Nothing in those sentences teaches or even suggests configuring a server program so that it listens for requests for secure hypertext transfer protocol sessions on port 80 (normally assigned to HTTP sessions) rather than port 443 (normally assigned to HTTPS sessions).

Appellant has diligently reviewed the remaining discussion of Ogdon's security subsystem, appearing in Ogdon at col. 16 line 32 to col. 18 line 19. Appellant has been unable to find in those portions of Ogdon anything that teaches or

even suggests configuring a server program so that it listens for requests for secure hypertext transfer protocol sessions on port 80 rather than port 443.

Ogdon Conclusion. Thus, for the reasons discussed above, Ogdon alone neither teaches nor suggests configuring a server program so that it listens for requests for secure hypertext transfer protocol sessions on port 80 rather than port 443, despite the Examiner's assertion to the contrary.

3.1.3. Scanlan and Ogdon combined do NOT teach or even suggest configuring a server program so that it listens for requests for secure hypertext transfer protocol sessions on port 80 rather than port 443. As discussed in detail above, neither Scanlan nor Ogdon teaches or even suggests the first element expressly required by claim 3, to wit: configuring a server program so that it listens for requests for secure hypertext transfer protocol sessions on port 80 rather than port 443. Consequently, even if for sake of discussion as of the effective filing date it had been possible to combine copies of both Scanlan and Ogdon into a single document, the resulting consolidated document would contain nothing that teaches or even suggests the first element expressly required by claim 3, to wit: configuring a server program so that it listens for requests for secure hypertext transfer protocol sessions on port 80 rather than port 443.

3.1.4. Scanlan and Ogdon cannot be combined. The effective filing date of the present application is at least as early as August 21, 1998. The Examiner's sole ground for the rejection of the pending claims is that, in the Examiner's view, on August 21, 1998 it would supposedly have been obvious to one skilled in the art to combine the teachings of Scanlan and Ogdon.

What the Examiner overlooks, however, is that on August 21, 1998, both Scanlan and Ogdon were secret within the United States Patent and Trademark Office. The patent application that issued as Scanlan was secret within the USPTO until February 22, 2000, when that patent was issued. The patent application that issued as Ogdon was secret within the USPTO until December 12, 2000, when that patent was issued.

It was thus physically impossible, in 1998, for anyone, whether skilled in the art or not, to combine the teachings of Scanlan and Ogdon. The earliest that anyone could have combined these references was in 2000.

For this reason, the rejection of the claims (having an effective filing date in 1998), over references which could not have been combined until 2000, must be reversed.

3.2. None of Scanlan, Ogdon, nor Scanlan combined with Ogdon, teaches or even suggests: receiving at the server program on port 80 a first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request is received on port 80 rather than port 443.

3.2.1. Scanlan alone does NOT teach or even suggest receiving at the server program on port 80 a first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request is received on port 80 rather than port 443. In the Office Action at page 2 line 22 to page 3 line 4, the Examiner (i) asserts that "Scanlan teaches … receiving at port 80 at the web server system a first data packet that is formatted in accordance with the protocol" and (ii) in support of this assertion (and multiple other assertions) cites Scanlan at Fig.1, Fig. 2, col.1 lines 46-60, col. 3 lines 1-6, and Fig. 4 (without stating which material supports which assertions). Taking these citations one at a time:

Scanlan at Fig. 1. In Fig. 1, titled "Web Security Injection Flow", all of the arrows between Scanlan's Web Server 12 and Scanlan's Browsers 10-1, 10-2, 10-3 and 10-4 point away from the server and towards the Browsers. Consequently, nothing in Scanlan's Fig. 1 teaches receiving anything at the server program on any port.

Scanlan at Fig. 2. Fig. 2 contains nothing about receiving, data packets, secure hypertext protocol or any port numbers.

Scanlan at col. 1 lines 46-60. These lines in Scanlan do at least contain the words "served" (which has the same root as "server"), "Secure Hypertext Transfer Protocol", "SHTTP", "Secure Sockets Layer", and "SSL". However, these lines do NOT mention port numbers at all.

Scanlan at col. 3 lines 1-6. These lines in Scanlan do at lest (i) contain the words "port" and "80", (ii) contain the word "request", which in the context of these lines probably implies receiving a data packet at a server and (iii) mention that ports other than 80 can be used for HTTP (i.e., unsecured) communications. However, these

lines do NOT teach or suggest receiving at the server program on port 80 a first data packet in a manner that is consistent with the secure hypertext transfer protocol.

Scanlan at Fig. 4. Fig. 4 does at least contain the word "message", which in the context of these lines probably implies a data packet. However, Fig. 4 contains nothing about ports and contains nothing about receiving data packets or messages. Consequently, these lines do NOT teach or suggest receiving at the server program on port 80 a first data packet in a manner that is consistent with the secure hypertext transfer protocol.

Scanlan Conclusion. Thus, for the reasons discussed above, Scanlan alone neither teaches nor suggests receiving at the server program on port 80 a first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request is received on port 80 rather than port 443, despite the Examiner's assertion to the contrary.

3.2.2. Ogdon alone does NOT teach or even suggest receiving at the server program on port 80 a first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request is received on port 80 rather than port 443. In the Office action at page 3 lines 7-9, the Examiner (i) asserts that "Ogdon teaches the port 80 to communicate with the server using the various protocols selected from the group consisting of: secure socket layer, secure sockets layer, SSL, secure hypertext transfer protocol, and HTTPS" and (ii) in support of this assertion cites Ogdon at col. 16 lines 14-31.

The cited lines in Ogdon do at least contain several relevant words (e.g., "security", "port numbers", "80", "servers", "HTTPS", "Secure Sockets Protocol"), but in those lines they are not connected together in a way that is relevant to this Application. See Ogdon at Col. 14 line 27 to col. 18 line 19 for the full context of the lines cited by the Examiner. See the discussion above at 3.1.2 for a summary of that discussion.

Turning now to the lines in Ogdon cited by the Examiner, and taking them one sentence at a time:

Ogdon at col. 16 lines 14-19 says:

> "Also note that the security measures for the present invention are not restricted to providing communications on generally used port numbers (e.g., communication between the host and leaders or audience members can occur on either port 60 or port 80 in any combination for a single presentation performance[])]."

Port 60 is normally unassigned. Thus, the above quoted words seem simply to reiterate the idea that using non-standard (e.g., normally unassigned) port numbers for web servers might enhance security. Nothing in the above quoted words teaches or even suggests receiving at the server program on port 80 a first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request is received on port 80 rather than port 443.

Ogdon at col. 16 lines 19 to 23 says:

> "Note that special security presentation performances can be run using any port number desired when using servers 248 in the operations center, or on intranets (e.g., the secure corporate intranet 260)."

The above quoted sentence simply reiterates the idea that using non-standard port numbers for web servers might enhance security. Nothing in the above quoted words teaches or even suggests using port 80 or any other port that is normally reserved for other protocols.

The next two sentences cited by the Examiner, Ogdon at col. 16 lines 23 to 32, discuss how the HTTPS protocol can be useful for transmitting questions to participants, collecting responses from participants and returning results to participants. Nothing in those sentences teaches or even suggests receiving at the server program on port 80 a first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request is received on port 80 rather than port 443.

Ogdon Conclusion. Thus, for the reasons discussed above, Ogdon alone neither teaches nor suggests receiving at the server program on port 80 a first data

packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request is received on port 80 rather than port 443.

3.2.3. Scanlan and Ogdon combined do NOT teach or even suggest receiving at the server program on port 80 a first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request is received on port 80 rather than port 443. As discussed in detail above, neither Scanlan nor Ogdon teaches or even suggests the second element expressly required by claim 3, to wit: receiving at the server program on port 80 a first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request is received on port 80 rather than port 443. Consequently, even if for sake of discussion as of the effective filing date it had been possible to combine copies of both Scanlan and Ogdon into a single document, the resulting consolidated document would contain nothing that teaches or even suggests the second element expressly required by claim 3, to wit: receiving at the server program on port 80 a first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request is received on port 80 rather than port 443.

3.2.4. Scanlan and Ogdon cannot be combined. As of the effective filing date of this application, it was physically impossible to combine the two references, for the reasons stated above in paragraph 3.1.4.

3.3. None of Scanlan, Ogdon, nor Scanlan combined with Ogdon, either teaches or suggests: outputting from the server program a response to the first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request was received on port 80 rather than port 443.

3.3.1. Scanlan alone does NOT teach or even suggest outputting from the server program a response to the first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request was received on port 80 rather than port 443. In the Office action at page 2 line 22 - page 3 line 4, the Examiner (i) asserts that "Scanlan Teaches … responding to the first data packet with a second data packet that is formatted in accordance with the protocol" and (ii) in support of this assertion (and multiple other assertions) cites Scanlan at Fig.1, Fig. 2, col.1 lines 46-60,

col. 3 lines 1-6, and Fig. 4 (without stating which material supports which assertions). Taking these citations one at a time:

Scanlan at Fig. 1. In Fig. 1, titled "Web Security Injection Flow", all of the arrows between Scanlan's Web Server 12 and Scanlan's Browsers 10-1, 10-2, 10-3 and 10-4 point away from the server and towards the Browsers. Consequently, nothing in Scanlan's Fig. 1 teaches a response to a first data packet received by the web server. Furthermore, nothing in Scanlan's Fig. 1 concerns port numbers.

Scanlan at Fig. 2. Fig. 2 contains nothing about outputting, server programs, responses, data packets, protocols or ports.

Scanlan at col. 1 lines 46-60. These lines in Scanlan do at least contain the words "served" (which has the same root as "server"), "Secure Hypertext Transfer Protocol", "SHTTP", "Secure Sockets Layer", and "SSL". However, these lines do NOT teach or suggest outputting from the server program a response to the first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request was received on port 80 rather than port 443. These lines do not mention port numbers at all.

Scanlan at col. 3 lines 1-6. These lines in Scanlan do at lest (i) contain the words "port" and "80" and (ii) mention that the ports other than 80 can be used for HTTP (i.e., unsecured) communications. However, these lines do NOT teach or suggest outputting from the server program a response to the first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request was received on port 80 rather than port 443.

Scanlan at Fig. 4. Fig. 4 contains nothing about port numbers. Consequently, Fig. 4 does NOT teach or suggest outputting from the server program a response to the first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request was received on port 80 rather than port 443.

Scanlan Conclusion. Thus, for the reasons discussed above, Scanlan alone neither teaches nor suggests outputting from the server program a response to the first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request was received on port 80 rather than port 443, despite the Examiner's assertion to the contrary.

3.3.2. Ogdon alone does NOT teach or even suggest outputting from the server program a response to the first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request was received on port 80 rather than port 443. In the Office action at page 3 lines 7-9, the Examiner (i) asserts that "Ogdon teaches the port 80 to communicate with the server using the various protocols selected from the group consisting of: secure socket layer, secure sockets layer, SSL, secure hypertext transfer protocol, and HTTPS" and (ii) in support of this assertion cites Ogdon at col. 16 lines 14-31.

The cited lines in Ogdon do at least contain several relevant words (e.g., "security", "port numbers", "80", "servers", "HTTPS", "Secure Sockets Protocol"), but in those lines they are not connected together in a way that is relevant to this Application. See Ogdon at Col. 14 line 27 to col. 18 line 19 for the full context of the lines cited by the Examiner. See the discussion above at 3.1.2 for a summary of that discussion.

Turning now to the lines in Ogdon cited by the Examiner, and taking them one sentence at a time:

Ogdon at col. 16 lines 14-19 says:

> "Also note that the security measures for the present invention are not restricted to providing communications on generally used port numbers (e.g., communication between the host and leaders or audience members can occur on either port 60 or port 80 in any combination for a single presentation performance[])]."

Port 60 is normally unassigned. Thus, the above quoted words seem simply to reiterate the idea that using non-standard (e.g., normally unassigned) port numbers for web servers might enhance security. Nothing in the above quoted words teaches or even suggests outputting from the server program a response to the first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request was received on port 80 rather than port 443.

Ogdon at col. 16 lines 19 to 23 says:

"Note that special security presentation performances can be run using any port number desired when using servers 248 in the operations center, or on intranets (e.g., the secure corporate intranet 260)."

The above quoted sentence simply reiterates the idea that using non-standard port numbers for web servers might enhance security. Nothing in the above quoted words teaches or even suggests outputting from the server program a response to the first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request was received on port 80 rather than port 443.

The next two sentences cited by the Examiner, Ogdon at col. 16 lines 23 to 32, discuss how the HTTPS protocol can be useful for transmitting questions to participants, collecting responses from participants and returning results to participants. Nothing in those sentences teaches or even suggests outputting from the server program a response to the first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request was received on port 80 rather than port 443.

Ogdon Conclusion. Thus, for the reasons discussed above, Ogdon alone neither teaches nor suggests outputting from the server program a response to the first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request was received on port 80 rather than port 443.

3.3.3. Scanlan and Ogdon combined do NOT teach or even suggest outputting from the server program a response to the first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request was received on port 80 rather than port 443. As discussed in detail above, neither Scanlan nor Ogdon teaches or even suggests the third element expressly required by claim 3, to wit: outputting from the server program a response to the first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request was received on port 80 rather than port 443. Consequently, even if for sake of discussion as of the effective filing date it had been possible to combine copies of both Scanlan and Ogdon into a single document, the resulting consolidated document would contain nothing that teaches or even suggests the third element expressly required by claim 3, to wit: outputting from the server program a response to the first data packet in a manner

that is consistent with the secure hypertext transfer protocol, except that the request was received on port 80 rather than port 443.

3.3.4. Scanlan and Ogdon cannot be combined. As of the effective filing date of this application, it was physically impossible to combine the two references, for the reasons stated above in paragraph 3.1.4.

3.4. At the time of Appellant's invention, the state of the art was that one seeking to use an alternate port number for a web server would be advised to use a port number other than 80 (the default port for HTTP).

As discussed in the Specification at paragraphs 232 – 236 (page 30, line 16 to page 31, line 11), Appellant's invention seeks to solve the following problem (the "Firewall Problem"): how can a client computer use HTTPS to communicate securely with a server computer when such client computer is connected to the Internet through a firewall that (i) blocks packets addressed to destination port 443 (the port number normally associated with HTTPS) but (ii) passes packets addressed to destination port 80.

At the time of Appellant's invention, the state of the art was that one seeking to use an alternate port number for a server would be advised to use a port number other than 80 (the default port for HTTP).

3.4.1. Apache manuals. The materials available at http://ftp.monash.edu.au/pub/ap/Apache/ch01.htm, http://ftp.monash.edu.au/pub/ap/Apache/ch04.htm, http://ftp.monash.edu.au/pub/ap/Apache/ch05.htm, http://ftp.monash.edu.au/pub/ap/Apache/ch07.htm (hereinafter, collectively, "Apache Manual") teach away from using port 80 for any protocol other than http and teach away from using any of ports 1 to 1024 as a nonstandard port for a server. These materials are included in the Evidence Appendix.

In Chapter 1 of Apache Manuals at Fig. 1.1 (page 3 of 11 when printed by Appellant), the www service (which uses http, the hyper text transfer protocol) is equated with port 80. This teaches away from the notion that port 80 should be used for other protocols, including without limitation SSL/https.

In Chapter 7 of Apache Manuals under the heading "Protecting Your Data from Outside Access" at "Caution" (which appears on page 29 of 38 when printed by Appellant), in the context of discussing how to hide a non-secure/http server, says in relevant part:

> "The second way to make your server less likely to be found is to run it on a nonstandard port. Ports can range from 0 to 65,535, so there is a wide range to choose from. Generally, the first 1024 are considered reserved ports."

By pointing out how many ports are potentially available and observing that the first 1024 are considered reserved ports, Apache teaches away from moving any server to a non-standard port in the range from 0 to 1024. That range includes port 80 which is normally associated with HTTP / hyper text transfer protocol.

3.4.2. Running a Perfect Web With Windows. Appellant previously filed a copy of Running a Perfect Web Site with Windows – Chapter 5, hereinafter "Windows (Chapter 5)" (from the web at http://www.gsu.unibel.by/pub/perf_web/06r07632.HTM). These materials are included in the Evidence Appendix.

The notice at the top of Windows (Chapter 5) says "Copyright © 1996" and is very similar to the notice at the top of Chapter 4 of Apache that was provided by the Examiner.

Windows (Chapter 5) at the bottom of page 3 says in relevant part:

> "… Ports under 1024 are reserved for the most common types of Internet traffic, so it is recommended that you use a number above 1024 if you need an alternate port. …"

3.5. Appellant's invention has unexpected, serendipitous or counter-intuitive results. At the time of Appellant's invention, based upon materials such as Apache Manuals and Windows (Chapter 5), one skilled in the art would have expected that changing the port number on which an HTTPS server listens for session requests would make it harder for clients to communicate with such server. However, for clients connected to the Internet through certain types of firewalls, configuring an HTTPS server to listen on port 80 can make it easier for a client to establish an HTTPS session with such server. In fact, in circumstances where it would NOT have been possible to

establish an HTTPS session with such server if it were listening on port 443, the default port for HTTPS, Appellant's invention makes an HTTPS session with such server possible.

It is unexpected, serendipitous and counter-intuitive that configuring a server to listen to a non-standard and unexpected port makes it easier for some clients to reach such server, since this is precisely the sort of change that Apache and Windows (Chapter 5) teaches will make it harder for browsers to communicate with such server.

The unexpected, serendipitous and counter-intuitive results obtained by practicing Appellant's invention cut strongly against the Examiner's view that Appellant's invention was obvious at the time it was made.

3.6. During the period (the "Post Ogdon Period") from December 12, 2000 (the later of the issue date of Scanlan and the issue date of Ogdon) through at least December 13, 2005 (the "Test Date"), a period of just over 5 years, others skilled in the art did NOT regularly and routinely duplicated Appellant's invention when confronted with the Firewall Problem (as defined above at 3.4).

3.6.1. General Discussion of the Failure of Others to Re-Invent Appellant's Wheel. December 13, 2005 is used as the Test Date because it is the last date as of which the behavior of amazon.com and barnsandnoble.com were both (i) observed by Appellant and (ii) reported to the Examiner in a filing with the Patent and Trademark Office. The materials reporting the results of the test are included in the Evidence Appendix.

The technical staff of every e-commerce web site that attempted to do business with the general public during the Post Ogdon Period ought eventually to have encountered the Firewall Problem since some customers and some potential customers ought to have spent some time at offices or other locations where their computers were connected to the Internet through firewalls that blocked outgoing packets addressed to destination port 443.

Consequently, if Appellant's invention should have been obvious to anyone skilled in the art who encountered (or continued to face) the Firewall Problem during the Post Ogdon Period, then it would be logical to expect that during the Post Ogdon Period the technical staffs of several e-commerce web sites that sought to do

business with the general public would either (i) have duplicated Appellant's invention (e.g., configured HTTPS servers to listen on port 80 and directed clients browsers to request information using a universal resource locator of the form "https://www.domain.com:80") or (ii) have settled upon some other solution to the Firewall Problem that permits secure communication with affected customers' computers.

However, Appellant is not personally aware of any web sites that, at any time (including, but not limited to, times during and after the Post Ogdon Period) have directed a customer's browser to a resource locator of the form https://www.domain.com:80 or implemented some other solution to the Firewall Problem that permits secure communication with customers' computers that are affected by the Firewall Problem (i.e., connected to the Internet through a firewall that passes outbound packets addressed to port 80 but blocks outbound packets addressed to port 443).

Consider for example the web sites barnesandnoble.com and amazon.com – two popular, highly competitive, technologically savvy e-commerce web sites that seek to conduct business with the general public.

3.6.2. barnesandnoble.com. It appears that when confronted with the Firewall Problem at times through the Test Date, the persons skilled in the art employed by barnesandnoble.com decided to drop back and punt. To ensure security, barnesandnoble.com used SSL (i.e., HTTPS) for order submission and the collection of credit card information. As of the Test Date, if a computer connected to the Internet through a firewall that blocked outgoing packets addressed to destination port 443 was used to attempt to place an order, the barnesandnoble.com web site would allow the user of such computer to fill up a shopping cart but at checkout time the browser on such computer would display an unhelpful error message as soon as the customer's browser was directed to establish an HTTPS session using the default destination port of 443. The materials reporting this test are included in the Evidence Appendix.

3.6.3. amazon.com. Based on tests conducted by Appellant on the Test Date, it is clear that when confronted with the Firewall Problem, the persons skilled in the art employed by amazon.com failed, at all times prior to and including the Test Date, to duplicate Appellant's invention or to implement some different solution that permits

encrypted communication with affected customers. On the Test Date, the folks at amazon.com clearly recognized the Firewall Problem, warned customers about it, and offered affected customers the choice of giving up or submitting order and payment details in an unsecured manner (i.e., using HTTP rather than HTTPS). In particular, on the Test Date, the checkout pages at amazon.com would send to a customer (using unsecure HTTP) a web page that contained both:

a button labeled:

"Sign in using our secure server"

and a link that said:

"The secure server will encrypt your information. If you received an error message when you tried to use our secure server, sign in using our standard server."

If on the Test Date a customer should have clicked on the button labeled "Sign in using our secure server", then such customer's browser would have been directed to a URL of the form "https://www.amazon.com/*", which by default implies destination port 443. If such customer's computer should have been connected to the Internet through a firewall that blocked outgoing packets addressed to destination port 443 and such customer should have clicked on such button, then such customer would have seen an uninformative error message.

If on the Test Date a customer should have clicked on the "standard server" link described above, then such customer's browser would have been directed to a URL of the form "http://www.amazon.com/*" which by default implies destination port 80, thereby avoiding part of the Firewall Problem. Unfortunately, since that URL begins with "http", the remainder of the checkout process, including the transmission of credit card information, would have been conducted using HTTP which is NOT encrypted for security.

3.6.4. Conclusion. Since popular e-commerce sites that sought to do business with the general public during the Post Ogdon Period, on the Test Date, and to the best knowledge of Appellant at times after the Test Date, neither (i) routinely use URLs of the form "https://www.securedomain.com/*:80" for the secure portions of their check out procedures nor (ii) routinely use some other solution to the Firewall Problem

that ensures secure, encrypted communications with affected customers' computers, Appellant contends that Appellant's invention was not obvious when it was first reduced to practice by Appellant, did not become obvious when Scanlan and Ogdon issued, and remains non-obvious today, more than eight (8) years later.

3.7. Ogdon teaches away from configuring a web server to use for a first protocol a port that is normally assigned to a second protocol.

Ogdon at col. 16 lines 14-19, the one place where Appellant found any disclosure by Ogdon of the particular non-standard port numbers that Ogdon prefers to assign to a server, Ogdon says:

> "Also note that the security measures for the present invention are not restricted to providing communications on generally used port numbers (e.g., communication between the host and leaders or audience members can occur on either port 60 or port 80 in any combination for a single presentation performance[])]."

Port 60 is unassigned, i.e., it is not generally used for any protocol. Port 80 is normally used for HTTP communications. Thus, where Ogdon might have made the Examiner happy (and Appellant unhappy) by suggesting that one associate HTTP with a port number which is normally associated with some other protocol, Ogdon in fact teaches away from the invention of Claim 3 by suggesting that one seeking security through obscurity ought to associate HTTP with port 60 (which is normally assigned to NO protocol).

3.8. The Examiner has not provided an adequate statement of the basis for the Examiner's view that the differences between the subject matter of Claim 3 on the one hand and the disclosure in Scanlan and/or Ogdon on the other hand, are such that the subject matter of Claim 3 as a whole would have been obvious at the time Appellant's invention was made to a person having ordinary skill in the art to which Claim 3 pertains. Without limiting the generality of the foregoing, the Office Action does not set forth or describe a set of modifications to Scanlan (both modifications suggested to the Examiner by Ogdon and other modifications suggested by the Examiner) that both (i) would bring Scanlan as so modified within the scope of Claim 3 and (ii) would have been obvious to one skilled in the art who was aware of Scanlan and/or Ogdon. Since the Examiner has

failed to provide a statement of the basis, if any, for the Examiner's view as to what one skilled in the art might do, Appellant argues that rejection for obviousness fails due to the absence, in the Office Action, of any reasonable statement as to the basis for the views expressed by the Examiner to provide the elements that are missing in the reference. See *In Re Alhert and Kruger*, 165 USPQ 418 (CCPA 1970).

In light of the foregoing, reversal is requested for Claim 3.

Argument - Claim 10

10. Claim 10 IS patentable over Scanlan in view of Ogdon because:

10.1. None of Scanlan, Ogdon, nor Scanlan combined with Ogdon, teaches or even suggests: configuring a web server system to use port 80 for communications using a protocol selected from the group consisting of: secure socket layer, secure sockets layer, SSL, secure hypertext transfer protocol, and HTTPS (such a protocol, a "Secure Protocol").

10.1.1. Scanlan alone does NOT teach or even suggest configuring a web server system to use port 80 for communications using a Secure Protocol. In the Office action at page 2 line 22 - page 3 line 4, the Examiner asserts that "Scanlan Teaches configuring the server program/system to use port 80 (rather than the first port number 443) for communications using a protocol". The portions of Scanlan cited by the Examiner in support of this assertion do NOT teach what the Examiner says they teach. See the detailed discussion above at 3.1.1.

10.1.2. Ogdon alone does NOT teach or even suggest configuring a web server system to use port 80 for communications using a Secure Protocol. In the Office action at page 3 lines 7-9, the Examiner asserts that "Ogdon teaches the port 80 to communicate with the server using the various protocols selected from the group consisting of: secure socket layer, secure sockets layer, SSL, secure hypertext transfer protocol, and HTTPS". However, the portions of Ogdon cited by the Examiner in support of this assertion do NOT teach what the Examiner says they teach. See the detailed discussion above at 3.1.2.

10.1.3. Scanlan and Ogdon combined do NOT teach or even suggest configuring a web server system to use port 80 for communications using a Secure Protocol. As discussed in detail above, neither Scanlan nor Ogdon teaches or even

suggests the first element expressly required by Claim 10, to wit: configuring a web server system to use port 80 for communications using a Secure Protocol. Consequently, even if for sake of discussion as of the effective filing date it had been possible to combine copies of both Scanlan and Ogdon into a single document, the resulting consolidated document would contain nothing that teaches or even suggests the first element expressly required by Claim 10, to wit: configuring a web server system to use port 80 for communications using a Secure Protocol.

10.1.4. Scanlan and Ogdon cannot be combined. As of the effective filing date of this application, it was physically impossible to combine the two references, for the reasons stated above in paragraph 3.1.4.

10.2. None of Scanlan, Ogdon, nor Scanlan combined with Ogdon, teaches or even suggests: receiving at port 80 at the web server system a first data packet that is formatted in accordance with the Secure Protocol.

10.2.1. Scanlan alone does NOT teach or even suggest receiving at port 80 at the web server system a first data packet that is formatted in accordance with the Secure Protocol. In the Office Action at page 2 line 22 to page 3 line 4, the Examiner asserts that "Scanlan teaches … receiving at port 80 at the web server system a first data packet that is formatted in accordance with the protocol". However, the portions of Scanlan cited by the Examiner in support of that assertion do NOT teach what the Examiner says they teach. See the detailed discussion above at 3.2.1.

10.2.2. Ogdon alone does NOT teach or even suggest receiving at port 80 at the web server system a first data packet that is formatted in accordance with the Secure Protocol. In the Office action at page 3 lines 7-9, the Examiner asserts that "Ogdon teaches the port 80 to communicate with the server using the various protocols selected from the group consisting of: secure socket layer, secure sockets layer, SSL, secure hypertext transfer protocol, and HTTPS". However, the portions of Ogdon cited by the Examiner in support of such assertion do not teach what the Examiner says they teach. See the detailed discussion above at 3.2.2.

10.2.3. Scanlan and Ogdon combined do NOT teach or even suggest receiving at port 80 at the web server system a first data packet that is formatted in accordance with the Secure Protocol. As discussed in detail above, neither Scanlan nor

Ogdon teaches or even suggests the second element expressly required by Claim 10, to wit: receiving at port 80 at the web server system a first data packet that is formatted in accordance with the Secure Protocol. Consequently, even if for sake of discussion as of the effective filing date it had been possible to combine copies of both Scanlan and Ogdon into a single document, the resulting consolidated document would contain nothing that teaches or even suggests the second element expressly required by Claim 10, to wit: receiving at port 80 at the web server system a first data packet that is formatted in accordance with the Secure Protocol.

10.2.4. Scanlan and Ogdon cannot be combined. As of the effective filing date of this application, it was physically impossible to combine the two references, for the reasons stated above in paragraph 3.1.4.

10.3. None of Scanlan, Ogdon, nor Scanlan combined with Ogdon, either teaches or suggests: responding to the first data packet with a second data packet that is formatted in accordance with the Secure Protocol.

10.3.1. Scanlan alone does NOT teach or even suggest responding to the first data packet with a second data packet that is formatted in accordance with the Secure Protocol. In the Office action at page 2 line 22 - page 3 line 4, the Examiner asserts that "Scanlan Teaches … responding to the first data packet with a second data packet that is formatted in accordance with the protocol". However, the portions of Scanlan cited by the Examiner to support this assertion do not teach what the Examiner says they teach. See the detailed discussion above at 3.3.1.

10.3.2. Ogdon alone does NOT teach or even suggest responding to the first data packet with a second data packet that is formatted in accordance with the Secure Protocol. In the Office action at page 3 lines 7-9, the Examiner asserts that "Ogdon teaches the port 80 to communicate with the server using the various protocols selected from the group consisting of: secure socket layer, secure sockets layer, SSL, secure hypertext transfer protocol, and HTTPS". However, the portions of Scanlan cited by the Examiner to support this assertion do not teach what the Examiner says they teach. See the detailed discussion above at 3.3.2.

10.3.3. Scanlan and Ogdon combined do NOT teach or even suggest responding to the first data packet with a second data packet that is formatted in

accordance with the Secure Protocol. As discussed in detail above, neither Scanlan nor Ogdon teaches or even suggests the third element expressly required by Claim 10, to wit: responding to the first data packet with a second data packet that is formatted in accordance with the Secure Protocol. Consequently, even if for sake of discussion as of the effective filing date it had been possible to combine copies of both Scanlan and Ogdon into a single document, the resulting consolidated document would contain nothing that teaches or even suggests the third element expressly required by Claim 10, to wit: responding to the first data packet with a second data packet that is formatted in accordance with the Secure Protocol.

10.3.4. Scanlan and Ogdon cannot be combined. As of the effective filing date of this application, it was physically impossible to combine the two references, for the reasons stated above in paragraph 3.1.4.

10.4. At the time of Appellant's invention, the state of the art was that one seeking to use an alternate port number for a web server would be advised to use a port number other than 80 (the default port for HTTP). See the detailed discussion of this point above at 3.4.

10.5. Appellant's invention has unexpected, serendipitous or counter-intuitive results. See the detailed discussion of this point above at 3.5.

10.6. During the Post Ogdon Period (from December 12, 2000 through at least December 13, 2005), a period of just over 5 years, others skilled in the art did NOT regularly and routinely duplicated Appellant's invention when confronted with the Firewall Problem (as defined above at 3.4). See the detailed discussion above at 3.6.

10.7. Ogdon teaches away from configuring a web server to use for a first protocol a port that is normally assigned to a second protocol. See the detailed discussion above at 3.7.

10.8. The Examiner has not provided an adequate statement of the basis for the examiner's view that the differences between the subject matter of Claim 10 on the one hand and the disclosure in Scanlan and/or Ogdon on the other hand, are such that the subject matter of Claim 10 as a whole would have been obvious at the time Appellant's invention was made to a person having ordinary skill in the art to which Claim 10 pertains. Without limiting the generality of the foregoing, the Office Action does not set

forth or describe a set of modifications to Scanlan (both modifications suggested to the Examiner by Ogdon and other modifications suggested by the Examiner) that both (i) would bring Scanlan as so modified within the scope of Claim 10 and (ii) would have been obvious to one skilled in the art who was aware of Scanlan and/or Ogdon. Since the Examiner has failed to provide a statement of the basis, if any, for the Examiner's view as to what one skilled in the art might do, Appellant argues that rejection for obviousness fails due to the absence, in the Office Action, of any reasonable statement as to the basis for the views expressed by the Examiner to provide the elements that are missing in the reference. See *In Re Alhert and Kruger*, 165 USPQ 418 (CCPA 1970).

In light of the foregoing, reversal is requested for Claim 10.

Argument - Claim 11

11. Claim 11 IS patentable over Scanlan in view of Ogdon because:

11.1. None of Scanlan, Ogdon, nor Scanlan combined with Ogdon, teaches or even suggests: web server software configured to use port 80 for communications using a Secure Protocol.

11.1.1. Scanlan alone does NOT teach or even suggest web server software configured to use port 80 for communications using a Secure Protocol. In the Office action at page 2 line 22 - page 3 line 4, the Examiner asserts that "Scanlan Teaches configuring the server program/system to use port 80 (rather than the first port number 443) for communications using a protocol". The portions of Scanlan cited by the Examiner in support of this assertion do NOT teach what the Examiner says they teach. See the detailed discussion above at 3.1.1.

11.1.2. Ogdon alone does NOT teach or even suggest web server software configured to use port 80 for communications using a Secure Protocol. In the Office action at page 3 lines 7-9, the Examiner asserts that "Ogdon teaches the port 80 to communicate with the server using the various protocols selected from the group consisting of: secure socket layer, secure sockets layer, SSL, secure hypertext transfer protocol, and HTTPS". However, the portions of Ogdon cited by the Examiner in support of this assertion do NOT teach what the Examiner says they teach. See the detailed discussion above at 3.1.2.

11.1.3. Scanlan and Ogdon combined do NOT teach or even suggest web server software configured to use port 80 for communications using a Secure Protocol. As discussed in detail above, neither Scanlan nor Ogdon teaches or even suggests the first element expressly required by Claim 11, to wit: web server software configured to use port 80 for communications using a Secure Protocol. Consequently, even if for sake of discussion as of the effective filing date it had been possible to combine copies of both Scanlan and Ogdon into a single document, the resulting consolidated document would contain nothing that teaches or even suggests the first element expressly required by Claim 11, to wit: web server software configured to use port 80 for communications using a Secure Protocol.

11.1.4. Scanlan and Ogdon cannot be combined. As of the effective filing date of this application, it was physically impossible to combine the two references, for the reasons stated above in paragraph 3.1.4.

11.2. None of Scanlan, Ogdon, nor Scanlan combined with Ogdon, teaches or even suggests: a means for receiving at port 80 at the web server system a first data packet that is formatted in accordance with the Secure Protocol.

11.2.1. Scanlan alone does NOT teach or even suggest a means for receiving at port 80 at the web server system a first data packet that is formatted in accordance with the Secure Protocol. In the Office Action at page 2 line 22 to page 3 line 4, the Examiner asserts that "Scanlan teaches … receiving at port 80 at the web server system a first data packet that is formatted in accordance with the protocol". However, the portions of Scanlan cited by the Examiner in support of that assertion do NOT teach what the Examiner says they teach. See the detailed discussion above at 3.2.1.

11.2.2. Ogdon alone does NOT teach or even suggest a means for receiving at port 80 at the web server system a first data packet that is formatted in accordance with the Secure Protocol. In the Office action at page 3 lines 7-9, the Examiner asserts that "Ogdon teaches the port 80 to communicate with the server using the various protocols selected from the group consisting of: secure socket layer, secure sockets layer, SSL, secure hypertext transfer protocol, and HTTPS". However, the portions of Ogdon cited by the Examiner in support of such assertion do not teach what the Examiner says they teach. See the detailed discussion above at 3.2.2.

11.2.3. Scanlan and Ogdon combined do NOT teach or even suggest a means for receiving at port 80 at the web server system a first data packet that is formatted in accordance with the Secure Protocol. As discussed in detail above, neither Scanlan nor Ogdon teaches or even suggests the second element expressly required by Claim 11, to wit: a means for receiving at port 80 at the web server system a first data packet that is formatted in accordance with the Secure Protocol. Consequently, even if for sake of discussion as of the effective filing date it had been possible to combine copies of both Scanlan and Ogdon into a single document, the resulting consolidated document would contain nothing that teaches or even suggests the second element expressly required by Claim 11, to wit: a means for receiving at port 80 at the web server system a first data packet that is formatted in accordance with the Secure Protocol.

11.2.4. Scanlan and Ogdon cannot be combined. As of the effective filing date of this application, it was physically impossible to combine the two references, for the reasons stated above in paragraph 3.1.4.

11.3. None of Scanlan, Ogdon, nor Scanlan combined with Ogdon, either teaches or suggests: a means for responding to the first data packet with a second data packet that is formatted in accordance with the Secure Protocol.

11.3.1. Scanlan alone does NOT teach or even suggest a means for responding to the first data packet with a second data packet that is formatted in accordance with the Secure Protocol. In the Office action at page 2 line 22 - page 3 line 4, the Examiner asserts that "Scanlan Teaches … responding to the first data packet with a second data packet that is formatted in accordance with the protocol". However, the portions of Scanlan cited by the Examiner to support this assertion do not teach what the Examiner says they teach. See the detailed discussion above at 3.3.1.

11.3.2. Ogdon alone does NOT teach or even suggest a means for responding to the first data packet with a second data packet that is formatted in accordance with the Secure Protocol. In the Office action at page 3 lines 7-9, the Examiner asserts that "Ogdon teaches the port 80 to communicate with the server using the various protocols selected from the group consisting of: secure socket layer, secure sockets layer, SSL, secure hypertext transfer protocol, and HTTPS". However, the

portions of Scanlan cited by the Examiner to support this assertion do not teach what the Examiner says they teach. See the detailed discussion above at 3.3.2.

11.3.3. Scanlan and Ogdon combined do NOT teach or even suggest a means for responding to the first data packet with a second data packet that is formatted in accordance with the Secure Protocol. As discussed in detail above, neither Scanlan nor Ogdon teaches or even suggests the third element expressly required by Claim 11, to wit: a means for responding to the first data packet with a second data packet that is formatted in accordance with the Secure Protocol. Consequently, even if for sake of discussion as of the effective filing date it had been possible to combine copies of both Scanlan and Ogdon into a single document, the resulting consolidated document would contain nothing that teaches or even suggests the third element expressly required by Claim 11, to wit: a means for responding to the first data packet with a second data packet that is formatted in accordance with the Secure Protocol.

11.3.4. Scanlan and Ogdon cannot be combined. As of the effective filing date of this application, it was physically impossible to combine the two references, for the reasons stated above in paragraph 3.1.4.

11.4. At the time of Appellant's invention, the state of the art was that one seeking to use an alternate port number for a web server would be advised to use a port number other than 80 (the default port for HTTP). See the detailed discussion of this point above at 3.4.

11.5. Appellant's invention has unexpected, serendipitous or counter-intuitive results. See the detailed discussion of this point above at 3.5.

11.6. During the Post Ogdon Period (from December 12, 2000 through at least December 13, 2005), a period of just over 5 years, others skilled in the art did NOT regularly and routinely duplicated Appellant's invention when confronted with the Firewall Problem (as defined above at 3.4). See the detailed discussion above at 3.6.

11.7. Ogdon teaches away from configuring a web server to use for a first protocol a port that is normally assigned to a second protocol. See the detailed discussion above at 3.7.

11.8. The Examiner has not provided an adequate statement of the basis for the examiner's view that the differences between the subject matter of Claim 11 on the one

hand and the disclosure in Scanlan and/or Ogdon on the other hand, are such that the subject matter of Claim 11 as a whole would have been obvious at the time Appellant's invention was made to a person having ordinary skill in the art to which Claim 11 pertains. Without limiting the generality of the foregoing, the Office Action does not set forth or describe a set of modifications to Scanlan (both modifications suggested to the Examiner by Ogdon and other modifications suggested by the Examiner) that both (i) would bring Scanlan as so modified within the scope of Claim 11 and (ii) would have been obvious to one skilled in the art who was aware of Scanlan and/or Ogdon. Since the Examiner has failed to provide a statement of the basis, if any, for the Examiner's view as to what one skilled in the art might do, Appellant argues that rejection for obviousness fails due to the absence, in the Office Action, of any reasonable statement as to the basis for the views expressed by the Examiner to provide the elements that are missing in the reference. See *In Re Alhert and Kruger*, 165 USPQ 418 (CCPA 1970).

In light of the foregoing, reversal is requested for Claim 11.

In light of the foregoing, reversal is requested for Claims 3, 10 and 11.

Respectfully,

Oppedahl Patent Law Firm LLC

/s/

Carl Oppedahl
PTO Reg. No. 32,746
P O Box 4850
Frisco, CO 80443-4850
telephone 970-468-8600

CLAIMS APPENDIX

Claims:

1. (Canceled)

2. (Canceled)

3. A method for securely communicating with a server program using a secure hypertext transfer protocol which by default uses a first port number associated therewith, said method practiced in connection with a hypertext transfer protocol which defaults to the use of a second port number associated therewith, said method comprising:

(a) configuring the server program so that it listens for requests for secure hypertext transfer protocol sessions on the second port number rather than the first port number;

(b) receiving at the server program on the second port number a first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request is received on the second port number rather than the first port number;

(c) outputting from the server program a response to the first data packet in a manner that is consistent with the secure hypertext transfer protocol, except that the request was received on the second port number rather than the first port number;

wherein the first port number is 443 and the second port number is 80.

4. (Canceled)

5. (Canceled)

6. (Canceled)

7. (Canceled)

8. (Canceled)

9. (Canceled)

10. A method for operating a web server system comprising:

(a) configuring the web server system to use port 80 for communications using a protocol selected from the group consisting of: secure socket layer, secure sockets layer, SSL, secure hypertext transfer protocol, and HTTPS;

(b) receiving at port 80 at the web server system a first data packet that is formatted in accordance with the protocol; and

(c) responding to the first data packet with a second data packet that is formatted in accordance with the protocol.

11. A web server system comprising:

(a) web server software configured to use port 80 for communications using a protocol selected from the group consisting of: secure socket layer, secure sockets layer, SSL, secure hypertext transfer protocol, and HTTPS;

(b) means for receiving at port 80 at the web server system a first data packet that is formatted in accordance with the protocol; and

(c) means for responding to the first data packet with a second data packet that is formatted in accordance with the protocol.

EVIDENCE APPENDIX

Chapter 01 - The State of the World Wide Web

In the six years since Tim Berners-Lee unleashed his graphical NeXT application, "WWW.app," upon an unsuspecting public, the World Wide Web has grown into *the* standard networked information infrastructure. (See **http://www.w3.org/pub/WWW/History.html**.) Its graphical interface and hypertext capabilities have caught the fancy of individuals and the media like no other Internet tool in history. Businesses, schools, government and nonprofit organizations, and millions of individuals are flocking to the Web to promote themselves and their products in front of an audience spanning the entire planet. Millions more are using the Web on a daily basis as a tool to conduct business, get informed, be entertained, and even form virtual communities.

It's difficult to watch a sporting event, a commercial, or even the news without seeing that increasingly familiar http:// - telling us of yet another enterprise on the Web. Because of the Web's popularity and its cost-effectiveness as a marketing tool, the World Wide Web is quickly becoming the electronic marketplace of the decade.

In this chapter, you learn:

- Where the Web has been
- How Web usage is changing
- How you might do business on the Web
- Where the Web is going

The Scope of the World Wide Web

The Web is now accessible in over 200 countries on all seven continents, and its information and services range from the esoteric

to the absurd. As of this writing, the AltaVista search engine reports that its robot has indexed 21 million Web pages, and the Netcraft Web Server Survey reports 135,000 different Web servers in its database. Web sites are maintained by universities, companies, public institutions, states, cities, and even high schools. Even McMurdo Station in Antarctica has a Web site. A number of powerful search engines (like AltaVista) and catalog sites (like Yahoo) allow rapid information location and retrieval, making the Web the ultimate tool for research, interactive entertainment, and even advertising. For more information, see **http://www.w3.org/pub/WWW/History.html**, **http://www.altavista.digital.com/**, **http://www.netcraft.co.uk/Survey/Reports**, **http://www.mcmurdo.gov/**.

One of several reasons why the World Wide Web rose to such prominence was because the underlying technology, HyperText Transfer Protocol (HTTP) and the Hypertext Markup Language (htmL), were "free." (See **http://www.w3.org/pub/WWW/Protocols/** and **http://www.w3.org/pub/WWW/MarkUp/**.)Anyone could write an htmL viewer or HTTP application without having to pay anyone royalties or licensing fees. This also made it easy for the Web to be platform independent - a Microsoft Windows Web browser has no problem talking to a UNIX Web server, and a Windows Web browser can display the htmL pages exactly the same as a Macintosh Web browser or UNIX Web browser does.

Furthermore, because htmL describes documents at a structural level rather than a "pretty picture" level, htmL is extremely portable between platforms of different capabilities. In other words, it is possible to write a well-formed htmL page that looks equally attractive on a graphical Web browser like Mosaic as it does on a text-only browser like Lynx. In fact, some companies are building audio-only Web browsers for the visually impaired, and htmL's structural markup makes this not only possible, but quite elegant.

NOTE

The Web is not free from forward compatibility problems - htmL has lacked a formal evolutionary strategy, and the one in HTTP (content negotiation) has not been widely implemented. Thus, many companies use new htmL tags, some of which cause older browsers to act inelegantly. This means you will often see sites that say "You must use Browser X to view these pages," which is more often a statement on the page author's capabilities than your browser's capabilities, since a site designed with care can be elegant for all browsers. Web standards coordination is the responsibility of the World Wide Web Consortium. See **http://www.w3.org/**.

There are many good sources for Internet usage statistics. One is located at Matrix Information and Directory Service (**http://www.mids.org/**). There are still statistics available at NSFNet (**http://nis.nsf.net/nsfnet/statistics/**), though the project was dismantled in April 1995.

The Web's Phenomenal Growth

In January 1993, there were only 50 known Web servers in existence. Today, the Web has become the largest source of traffic on the Internet. Table 1.1 and figure 1.1 show the growth of the Web relative to other Internet services on the Internet. You can find more details about the data shown in figure 1.1 at **http://www.nielsenmedia.com/demo.htm**. As was mentioned earlier, Web servers are in almost every developed country in the world.

NOTE

If you're interested, a list of all registered servers is available from **http://www.w3.org/hypertext/DataSources/WWW/Servers.html**.

Fig. 1.1 - This data was taken by a Nielsen survey conducted on Web usage.

Table 1.1 Growth of World Wide Web Traffic

Service Name	Port	Rank %	Pkts	Rank %	Bytes
ftp-data	20	1	18.758	1	30.251
www	80	2	13.122	2	17.693
telnet	23	3	10.357	6	3.715

Service Name	Port	Rank %	Pkts	Rank %	Bytes
www	80	1	21.443	1	26.250

ftp-data	20	2	14.023	2	21.535
nntp	119	3	8.119	3	8.657

Percentage of Total Byte Traffic Change on the NSF Backbone in a Four Month Period

As table 1.1 illustrates, the World Wide Web already comprises more traffic than any other Internet function. Despite the fact that the Web has been in operation for several years now, it is still able to grow at a rate of almost 20 percent a year.

NOTE

The World Wide Web traffic in table 1.1 reflects only connections to World Wide Web servers. Web browsers can also connect to FTP (File Transfer Protocol), Gopher, and other types of servers.

But do we know anything else about who is actually on the Web? The Nielsen study mentioned in figure 1.1 tells us quite a bit about who is on the Net. Among the findings:

- 56 percent of WWW users were between 25-44 years old.
- 64.5 percent of users were male.
- 88 percent had at least some college education.

The Proliferation of Web Server Software

One of the barometers of the growth of the Web is the incredible number of different products out there, particularly in the Web server field. Two years ago, there were roughly half a dozen Web servers, all products of research groups or experimentation. Now there are dozens and dozens of different Web server products out there. Many are commercial, but the free Web servers (like Apache) continue to be developed, supported, and very widely used. The chart available from WebCompare shows over 45 actively supported Web servers. For more information, see **http://www.webcompare.com/server-main.html**.

Why You Want To Be on the WWW

There is little doubt that there are some huge benefits to being on the Web today. In business, numbers speak volumes. There is no doubt that the Web has them. Web users are generally educated,

professional middle to upper-middle class people who want to use the Web for information, research, fun, and even for purchasing products.

Like every new major medium that has come before, the Web has distinct, inherent, and unique advantages to other media. Instant access to information resources - also known as the "pull" model compared to television's "push" model - is one of the most significant advantages. Many commercial sites report thousands of visitors within the first days of operation. Electronic malls are appearing everywhere, and financial transactions are becoming safer all the time.

The best thing about the Web, of course, is that it isn't going to go away. It's only going to get bigger and bigger. Connections will get faster, computers will get better, programming will get slicker, and access will get better. Most importantly, more people will getting online.

So, you're convinced. The Web is the greatest thing since tail fins, right? Well, almost. There are definitely a lot of advantages to doing business on the Net (as well as some pitfalls), and it will definitely be helpful to know about some of them. Who's out there? What are they like? Are they ready to buy your product? Who's doing business on the Web?

Some of those questions are easy to answer. We know that there are a lot of educated professionals on the Internet. We also know that many of them are involved in education, research, and industry. It's time to dig a little deeper and find out a little bit about how the Web can serve businesses and consumers of all kinds.

More Than a High-Tech Billboard: Your Name on the Web

The Web has proven that people will come - in droves - to the Internet if it's easy to use and accessible. For those in business for profit, being on the Web, referred to as "having a presence on the Web," usually serves two main purposes.

One reason many business get on the Web is to sell their product. Many companies, such as CD-Now, are focused on marketing a specific product or class of products (see fig. 1.2). They are not as concerned about establishing a brand identity or giving information away. They offer a product, and hope people will buy it. For more information about CD-Now, see **http://www.cdnow.com/**.

Fig. 1.2 - CD-Now is a Web-based company that sells music on the Internet.

Another big motivation for businesses to get on the Net is for advertising purposes. These companies want to further the equity they have established with their name brand, whether it be local, statewide, or national (see figures 1.3 and 1.4).

Fig. 1.3 - GISD is a Michigan company that provides Internet training and other services.

Fig. 1.4 - Even Coca-Cola, known the world around, advertises on the Web.

Companies who use the Web for this purpose are often service-based businesses, such as Global Information Services & Design in Michigan. Still others offer products that are just very difficult to sell over the Web and for whom product familiarity is of utmost importance. Again, these types of companies are generally national in nature or are service providers of some sort.

Table 1.2 shows what businesses reported when asked what they used the Web for. As you can see, many of the functions already being employed through other media are being utilized on the Web even today.

Table 1.2 Business Usage of the Web

Collaborate with others	54%
Publish information	33%
Gather information	77%
Research competitors	46%
Sell products or services	13%
Purchase products or services	23%
Provide customer service and support	38%
Communicate internally	44%
Provide vendor support and communications	50%

Percent of WWW Business Users Who Have Used It to...

Web Demographics: Who Is Your Audience?

It's time to talk about a few specifics about who exactly is on the Internet, and whether they actually buy what a business has to sell. We're going to return to the Nielsen survey mentioned in figure 1.1 for some more statistics that were gathered from 280,000 telephone interview nationwide.

If you're interested in getting a copy of the full report, The Final Report is available for purchase from CommerceNet (phone: 415-617-8790; e-mail: **survey@commerce.net**) and Nielsen Media Research (phone: 813-738-3125; e-mail: **interactive@nielsenmedia.com**). You can also find summary information on the Web at **http://www.nielsenmedia.com/demo.htm**.

So what did the Nielsen survey find? Well, over 2.5 million Americans have purchased products and services over the WWW. Again, as with all other numbers, these too will continue to grow. Earlier in the chapter, you were given a glimpse of some general demographics of users. The survey showed specific results important for businesses.

- 25 percent of WWW users had incomes over $80,000 a year.
- 55 percent of users have used the Web to research products or services and 14 percent have actually purchased them.
- There was a user base of 18 million Web users in the United States and Canada
- Total time spent on the Internet in the U.S. and Canada was actually equivalent to the total time spent watching rented video-tapes!

Cautionary Note

As rosy a picture as the Web paints, there are some downsides. The biggest is that any Internet survey or usage statistics fail to take into account the still large majority of people who do not access the Internet. Even with 18 million users using the Web on a regular basis according to the Nielsen report, that still leaves over 250 million people in the U.S. and Canada who still aren't on, not to mention the billions of people around the world who have yet to get online, where the rate of Internet penetration is even less. See **http://www.census.gov/**and **http://www.statcan.ca/Documents/English/Faq/Pop/pop.htm**.

The Internet is not yet (nor will it likely ever be) a panacea for everyone's advertising and marketing woes. It's another tool that can, and should, be utilized along with other more traditional media.

What Will the Web Be Like Tomorrow? Next Week?

Now that you have a better idea of where the Web has been and where it is, wouldn't you like to know where it's going? Wouldn't we all? A popular TV commercial shows all sorts of fanciful futuristic gadgets as being "the future." The commercial ends with the conclusion that each possibility is likely and it's sheer guesswork as to what the future will actually hold. To an extent, that commercial is right, but we can make some educated guesses.

We know that many advances are being made in technology that are now used on the cutting edge. Although we can't know exactly what everything will be like later, we can attempt to point out some directions the Web appears to be moving in, and what in particular you should be thinking about.

Problems with Today's Web Technology

There's no doubt the Web's popularity has benefited in no small part from increased public awareness and the availability of dial-up Internet connections. But, let's face it, if any of you have tried to look at a complicated Web site using a 28.8 kbps modem, you know that we've still got a long way to go.

Not only are there problems with access speed, but, as was mentioned in the last section, a large segment of the population remains untapped. The culture of the Internet is also changing - as more people get online, the demographics shift from those primarily in the computer and academic industry, to something that reflects more of the mainstream American and worldwide culture. This is on the whole a very good thing, but it can lead to some transitional problems, as we will later see.

Finally, a big roadblock to online commerce is the lack, or perceived lack, of security on the Internet. This problem is partially technological and partially psychological. There are protocols that can encrypt and validate transactions, such as Netscape's "Secure Sockets Layer" protocol or TERISA system's "SHTTP" protocol. (See **http://home.netscape.com/newsref/std/SSL.html** and **http://www.terisa.com/shttp/intro.html**.)But many users are concerned about giving, say, their credit card to an entity they only know about through the Internet. A few well-publicized Internet hacking incidents have also discouraged trust. This will be solved, but not by technology alone.

Breaking the Speed Limits

In the past, getting a full connection to the Internet required a high-

speed leased telephone line and expensive networking hardware. As a result, only businesses and large institutions could afford Internet access. This limited the Internet's usefulness for commercial purposes. However, the introduction of high-speed modems and dial-up Internet Service Providers (ISPs) has made WWW access from home both possible and practical.

The Serial Line Internet Protocol and Point-to-Point Protocol (SLIP and PPP) are two commonly used schemes for transferring Internet data to a home computer over the regular phone system. These protocols allow home users to obtain full Internet connections without having to purchase a leased line or expensive connecting hardware. This means that SLIP and PPP users can do everything that users with faster leased-line connections can do, albeit quite a bit slower.

NOTE

"SL/IP" is equivalent to "SLIP." Both refer to Serial Line Internet Protocol; this book uses SLIP.

But as services get bigger and more complicated, even high-speed modems often don't do the job. The use of ISDN (Integrated Services Digital Network) lines has recently become more popular, but even this solution brings up the problem of needing specialized add-on cards and protocols. ISDN is also quite expensive and is not available in many areas. As an example, a typical ISDN line in North Carolina now costs over $200 for installation and will cost an additional $75 per month to maintain (for more on ISDN, go to **http://alumni.caltech.edu/~dank/isdn/**).

Two areas that seem to hold a lot of promise for solving the bandwidth problems are cable modem access and satellite delivery. Satellite delivery is probably farther away, but some cable companies in the United States are already offering Internet access through the same line through which you receive your TV stations. One example is TCI in East Lansing, who already offer 10Mbps/sec (Ethernet speed) Internet connections for under $50 a month. It's expected that these types of connections will only get cheaper and more widespread in the future.

As more and more schools, libraries, community colleges and other public institutions get connected, those who use these facilities will also become Internet users. In addition, ISP rates will continue to fall and, as Internet Service becomes available through more accessible and accepted means (such as cable modems), people's fear of technology will also continue to decrease.

One of the last factors involved in increased usage will stem from a not-so-obvious source. In the past, if you wanted Internet service, you had to contact the provider, install the software, make the connection, and basically go through a lot of trouble to get online. However, with the breakout of Windows 95, OS/2 Warp and other "Internet-Ready" operating systems, the Internet is now built-in (see fig. 1.5). When Internet access becomes as easy as buying your computer, plugging it in, and getting online, a large barrier to access will be removed.

Fig. 1.5 - Microsoft's Internet Explorer incorporates the same functionality as Netscape right out of the box.

New Technologies

Web site developers and Web content creators are now getting faced with a dizzying array of new technologies: Java, VRML, Shockwave, MPEG audio and video, and more. (See **http://java.sun.com/**, **http://www.vrml.org/**, **http://www.macromedia.com/**.)and The Web has always been multimedia, but until recently that has been limited to inlined gif and JPEG images, and externally-played sound and movie files. As the state of browser technology has advanced, so too have the types of media that can be supported. Newer browsers that support a plug-in architecture can now support an arbitrary number of new media types, so we can expect to see an explosion of new types as companies start building these plug-ins.

Java and VRML are not covered in this book too deeply; they both are complex enough and powerful enough to merit their own books, and there are plenty of those out there. But well-prepared Webmasters should be aware of their existence, when they are appropriate, and how to integrate them into the server.

What Looks Good About the Web's Future

The future really is bright. We've already looked at many of the things that are available or soon will be that will make using the Web more efficient, profitable, and sensible. Perhaps one of the biggest benefits of all these changes is in the opportunity presented to small organizations without a lot of computer expertise or a lot of money to establish a presence on the Web.

Running a Web server, as this book will hopefully prove, can be not only a pretty rewarding experience, but also a pretty inexpensive endeavor. The software provided with this book is free. The software and hardware for a UNIX operating system can be pretty cheap if you purchase a 486 or Pentium and install Linux on it, and bandwidth is getting less expensive all the time. One reason for the success of the Web has been that it has been very easy to set up and

add content to a server, and thus there were no restrictions as to who could do it or what they could say. Even as "the big boys" come to play in this sandbox, that liberating capability is not likely to disappear.

As an example of this, the Windows 95 Web site (developed by a company other than Microsoft, by the way), at **http://www.windows95.com/**, runs BSDI on a Pentium with Apache (the same software provided with this book) and handles approximately 2 million hits on a busy weekday. The total hardware and software cost is somewhere around $5000, so don't let anyone fool you into thinking you need big expensive iron to put out a "real" Web site.

The Web has experienced terrific growth in the first several years of existence. Fueled by applications in business, government, education, and research, and turbo-charged by dramatic improvements in browser and server technology, the Web is poised to become *the* electronic marketplace and information source of the century.

 Table of Contents

 02 - Introduction to Web Servers

Chapter 04 - Getting Started with Your Web Server

While this chapter, like many others, is specific to the Apache server, the vocabulary is certainly applicable to other Web servers. In particular, the NCSA family of servers has much in common with Apache with respect to configuration files, since Apache was derived originally from the NCSA 1.3 server, and maintaining backwards compatibility with existing NCSA servers was a mandate with the development team.

This chapter deals with all the essential steps between the software on the CD-ROM (or downloaded from the Net) and a running, breathing, living server. If you have installed an Apache or NCSA server before, you can probably safely skip this chapter - perhaps skim it to look for essential differences. This chapter covers:

- The compilation of the source code for the server.
- The elemental parts of the configuration files.
- Some basic "why didn't the Web server come up?" debugging help.
- Obtaining, installing, and launching Apache-SSL.

Compiling Apache

Apache is known to compile on just about every UNIX variant out there: Solaris 2.X, SunOS 4.1.X, Irix 5.X and 6.X, Linux, FreeBSD/NetBSD/BSDI, HPUX, AIX, Ultrix, OSF1, NeXT, Sequent, AUX, SCO, UTS, Apollo Domain/OS, QNX, and probably a few you've never even tried yet. A port to OS/2 has been done, and a Windows NT port is rumored to be in the works. Portability has been a high priority for the development team.

Before you go about compiling Apache, make sure a binary suitable for your platform is not already available on the CD-ROM included with this book. There is a README file on the CD-ROM that explains which system binaries are provided. If there is a suitable binary, you can skip the compilation process and move on to the next section, although if you ever want to add new modules or tweak the functionality provided by Apache, you'll need to know how to compile it.

Copy the source code package to a part of your file system. Depending on which OS you are on, you'll need up to 10 spare megabytes of disk to compile the server. Unpack it and go to the /src subdirectory. A sequence of commands to do this might look like

```
cd /CDROM
cp apache_1.0.3.tar /usr/local/etc/
tar -xvf apache_1.0.3.tar
cd apache_1.0.3/src
```

Step 1: Edit the "Configuration" File

This file is used by the "Configure" program to create a "Makefile" specifically targeted to your platform, with any runtime defines set if necessary, and with the modules you have chosen compiled together. It also creates a "modules.c," which contains information about which modules to link together at compilation time.

You must declare which C compiler you are using, and you must uncomment the appropriate setting for "AUX_CFLAGS" and "AUX_LIBS" for the platform on which you are compiling. For example, the following is appropriate if you are using the GNU C compiler

```
CC=gcc
```

And if, say, you want to set it to use Solaris instead of the default, which is SunOS, you want to change the section which reads:

```
# For SunOS 4
AUX_CFLAGS= -DSUNOS4
# For Solaris 2.
#AUX_CFLAGS= -DSOLARIS2
#AUX_LIBS= -lsocket -lnsl
```

to the following:

```
# For SunOS 4
#AUX_CFLAGS= -DSUNOS4
# For Solaris 2.
AUX_CFLAGS= -DSOLARIS2
AUX_LIBS= -lsocket -lnsl
```

For the CFLAGS definition: if you want every file with the execute bit set to be parsed for server side includes, set "-DXBITHACK." If you wish to eliminate the overhead of performing the reverse-DNS lookup when an entry is written to the logfile, set "-DMINIMAL_DNS."

If, on the other hand, you want to have an even greater sense of confidence in the hostname, you can set "-DMAXIMAL_DNS." You would set this if you were protecting parts of your site based on hostname. Doing this is optional, and is mostly provided for backwards compatibility with NCSA 1.3.

At the bottom of the file is a list of packaged modules that come with the Apache distribution. Notice that not all of them are compiled in by default. To include a module in the build, uncomment the entry for it. Notice that some modules are mutually exclusive - for example, it would not be wise to compile both the configurable logging module and the common logging

module at the same time.

Also, some modules, like mod_auth_dbm, may require linking to an external library, and need an entry added to the EXTRA_LIBS line. You'll learn more about modules in a little bit, for the purposes of getting up and running I'd recommend simply using the defaults as provided.

Step 2: Run the "Configure" Script

This is a simple Bourne shell script that takes the "Configuration" file and creates a "Makefile" out of it, as well as "modules.c." If you are feeling ambitious, you can look at and edit "httpd.h," which sets a lot of defaults for some low-level functionality, most of which is set anyway in the configuration files.

Step 3: Run "make"

This compiles the server. You might see some warnings about data types, particularly if you compiled with -Wall set, but none of the errors should be fatal.

If all went well, you should now have an executable program in your /src directory called httpd.

Establishing the File Hierarchy

The next step in the process of setting up a server is to make some fundamental decisions regarding where on the file system different parts of the server will reside. Write down your decisions for each of these; they will be needed in the section on "Basic Configuration."

First, there is the *server root*. This is the subdirectory in which you unpacked the server, and from which the conf/ directory, the logs/ subdirectory, the cgi-bin/ subdirectory, and other server-related directories lead. The default suggestion is to have this as /usr/local/etc/httpd. You will be able to have your configuration files and log files in other locations - the server root was designed to be a convenient place to keep everything server-related together. Also, if the server crashes and leaves a core file, it will be found in the server root directory.

Second, there is the *document root*. This is the directory in which all your htmL and other media reside. A file in here called myfile.html would be referenced as http://host.com/myfile.html. It is recommended that this be outside the server root and in its own directory. Since you'll be referring to it frequently, it should be pretty short - for example, /home/www or /www/htdocs. If you are implementing a Web server on top of an FTP server, for example, you might want to point the document root at /home/ftp/pub.

Finally, you need to decide where on your server you will keep your logfiles. This should be a space with a fairly large working area, depending on how busy you estimate your server will be. For a point of reference, a site with 100K hits per day (which would fall under moderate traffic, relatively speaking) can expect to generate 15 MB per day of logfile information.

Later in this book, you'll deal with automated logfile rotation and logfile analysis tools, but for now just be aware of the disk space issue. Furthermore, for performance reasons, it's usually best to have the log directory on a separate disk partition or even a separate disk altogether,

since on even a moderately busy server the access log can be written to several times per second.

Basic Configuration

This section covers the minimal set of changes you need to make to the configuration files in order to launch a basic Web site.

See Chapter 6, "Managing an Internet Web Server," for advanced configuration.

There are three separate configuration files in Apache. This model goes back to NCSA, and the reasoning is sound: there are largely three main areas of administrative configuration, so setting them up as separate files allows the Web master to give different write permissions to each if he or she so desires.

You will find the configuration files for Apache in the conf/ subdirectory of the server root directory. Each has been provided with a "-dist" appendage; it is recommended that you make a copy without the "-dist" and edit those new files, keeping the "-dist" versions as backups and reference.

The basic format of the configuration files are a combination of a shell-like interface and pseudo-htmL. The elemental unit is the *directive*, which can take a number of arguments. Essentially:

```
Directive argument argument....
```

i.e.

```
Port 80
```

or

```
AddIcon /icons/back.gif ..
```

Directives can also be grouped together inside certain pseudo-htmL "tags." Unlike htmL, these tags should be on their own line. For example:

```
<Virtualhost www.myhost.com>
DocumentRoot /www/htdocs/myhost.com
ServerName www.myhost.com
</Virtualhost>
```

httpd.conf

The first configuration file to look at is "httpd.conf." This is the file that sets the basic system-level information about the server - what port it binds to, which users it runs under, and so on. If you are not the systems administrator of the site at which you are installing the server, you might want to ask the administrator to help you with these questions.

The essential items in this file to cover are:

```
Port <number>
```

for example,

```
Port 80
```

This is the TCP/IP port number to which the Web server binds. Port 80 is the default port in "http:" URLs; in other words, "http://www.myhost.com/" is equivalent to "http://www.myhost.com:80/."

For a number of reasons, however, you might want to run your server on a different port; for example, there is already a server running on port 80, or this is a server you want to keep "secret." (Though if there is sensitive information on this, you should at least do host-based access control, if not password protection.)

```
User <#number or uid>
Group <#number or uid>
```

as in

```
User nobody
Group nogroup
```

This is the UNIX user that the Web server will run as. Apache needs to be launched as root in order to bind to a port lower than 1024 - this is a basic security feature of all UNIX implementations. Immediately after "grabbing" the port, Apache changes its effective user ID to something else, typically as user "nobody." This is for security reasons - running your Web servers as root means that any hole in the server (be it through the server itself, or through a CGI script, which is much more likely) could be exploited by an outside user to run a command on your machine. Thus, setting the user to "nobody," "www," or some other reasonably innocuous user ID is the safest bet. This user ID needs to be able to read files in the document root, as well as have read permission on the configuration files. The argument should be the actual user name - if you want to give the numeric user ID, prepend the number with a pound sign (#). The Group directive is the same issue; decide which group ID you want the server to run with.

```
ServerAdmin <email address>
```

This should be set to the e-mail address of a user who can receive mail related to the actions of the server. In the case of a server error, the message given to the browser visiting your site will include a message to the effect of "please report this problem to user@myhost.com." In the future, Apache may send warning e-mail to this user if it encounters a major systems-related problem.

```
ServerRoot <directory>
```

for example

```
ServerRoot /usr/local/etc/httpd
```

This is the server root decided upon earlier. Give the full path, and don't end it with a slash.

```
ErrorLog <directory/filename>
TransferLog <directory/filename>
```

These two directives specify exactly where to log errors and Web accesses. If the filename given doesn't start with a slash, it is presumed to be relative to the server root directory. It was suggested earlier that the logfiles be sent to a separate directory outside of the server root; this is where you specify that logging directory and the name of the log files within that directory.

```
ServerName <DNS hostname>
```

At times, the Web server will have to "know" the hostname it is being referred to as, which can be different from its real hostname. For example, the name "www.myhost.com" might actually be a DNS alias for "gateway.myhost.com." In this case, you don't want the URLs generated by the server to be "http://gateway.myhost.com/." ServerName allows you to set that precisely.

srm.conf

The second configuration file to cover before launch is srm.conf. The important things to set in that file are:

```
DocumentRoot <directory>
```

As described before, this is the root level of your tree of documents - be that "/usr/local/etc/httpd/htdocs" or "/www/htdocs." Based on my experience, it's a very good idea to keep it short and concise. This directory must exist and be readable by the user the Web server runs as.

```
ScriptAlias <request path alias> <directory>
```

ScriptAlias lets you specify that a particular directory *outside* of the document root can be aliased to a path in the request, *and* that objects in that directory are executed instead of simply read from the file system. For example, the default offering

```
ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-bin/
```

means that a request for http://www.myhost.com/cgi-bin/fortune will execute the program /usr/local/etc/httpd/cgi-bin/fortune. Apache comes bundled with a number of useful beginner CGI scripts, simple shell scripts that illustrate CGI programming. However, you probably don't want to turn them by default. I recommend commenting this line out until you're sure you want to use it as your CGI invocation mechanism.

Finally, the directory containing the CGI scripts should *not* be under the document root - bizarre interactions between the code that handles "scriptalias" and the code that handles request/pathname resolution could cause problems.

Just as with httpd.conf, there are many extra features that are discussed in upcoming sections.

access.conf

Access.conf is structured more rigidly than the other configuration files; all the content is contained within <Directory></Directory> pseudo-htmL tags that define the scope of the

directives listed within. So for example, the directives in in between

```
<Directory /www/htdocs>
```

and

```
</Directory>
```

affect everything located under the /www/htdocs directory. Furthermore, wildcards may be used, for example

```
<Directory /www/htdocs/*/archives/>
....
</Directory>
```

applies to /www/htdocs/list1/archives/, /www/htdocs/list2/archives/, and so on. The most important directive to set at this point is "Options." "Options" takes a list of keywords that enable or disable particular functionality.

It's important to establish a conservative set of functionality when the site is first launched. I would recommend using just "Indexes" at the very beginning. For example:

```
Options Indexes
```

Starting Up Apache

To start Apache, simply run the binary you compiled earlier (or your precompiled binary) with the "-f" flag pointing to the httpd.conf file also created earlier. For example:

```
/usr/local/etc/httpd/src/httpd -f
    /usr/local/etc/httpd/conf/httpd.conf
```

It's probably a good idea at this point to use the UNIX command "ps" to see if "httpd" is running, typically something like

```
ps -augwx | grep httpd     (BSD-based systems)
ps -ef | grep httpd        (SVR4-based systems)
```

will suffice. To your surprise you, you will hopefully see a number of simultaneous "httpd" processes running. What's going on?

The first Web servers, like the CERN and NCSA servers, used the model of one main Web server "cloning" itself with every single request that came in. The "clone" would respond to the request, while the original server returned to listening to the port for another request. While certainly a simple and robust design, the act of "cloning" (or in UNIX terms, "fork") was an expensive operation under UNIX, so loads above a couple hits per second were quite punishing even on the nicest hardware. It was also difficult to implement any sort of "throttling," reducing the amount of "cloning" that took place when the number of "clones" was very high since it was hard for the original server to know how many "clones" were still around. Thus servers had no easy way to refuse or delay connections based on a lack of resources.

Apache, like NCSA 1.4+, Netscape's Web servers, and a couple of other UNIX-based Web servers, instead uses the model of a group of persistent "children" running in parallel. The children are coordinated by a "parent" process, which can tell how many children are alive, spawn new children if necessary, and even terminate old children if there are many idle ones, depending on the situation. *Parent* and *child* are the actual UNIX terms.

Back to the server. Fire up your Web browser and go to http://www.myhost.com/. Did it work? If all went well, you should be able to see a directory index listing of everything in the document root directory, or if there's an "index.html" in that directory, you would see the contents of that file.

Other command line options are shown in the following table:

Option	Result
-d serverroot	Sets the initial value for ServerRoot.
-X	Runs the server in single-process mode; useful for debugging purposes, but don't run the server in this mode for v serving content to the outside world.
-v	Prints the version of the server, and then exits.
-?	Prints the list of available command-line arguments to Apache.

Debugging the Server Start-Up Process

Apache is usually pretty good about giving meaningful error messages, but some are explained in more detail here.

```
httpd: could not open document config file .....
fopen: No such file or directory
```

This is usually the result of giving just a relative path to the -f argument, so Apache looks for it relative to the compiled-in server root (what's set in src/httpd.h) instead of relative to the directory you are in. You must give it either the full path or the path relative to the compiled-in server root.

```
httpd: could not bind to port [X]
bind: Operation not permitted
```

This was most likely caused by attempting to run the server on a port below 1024 without launching it as "root." Most UNIX operating systems prevent people without root access from trying to launch a server - any type of server - on a port less than 1024. If you launch the server as root, the error message should disappear.

```
httpd: could not bind to port
bind: Address already in use
```

This means that there is already something running on your machine at the port you have specified. Do you have another Web server running? There is no standard UNIX mechanism for determining what's running on what ports; on most systems, the file "/etc/services" can tell you what the most common daemons are, but it's not a complete list. You could also try using the "netstat" command, with various options such as "-a".

```
httpd: bad user name ....
httpd: bad group name ....
```

The "User" or the "Group" you had set in httpd.conf didn't actually exist on your system. You might see errors telling you that particular files or directories don't exist. If it looks like the files are there, make sure they are readable by the user IDs that the server runs as (i.e., both root and nobody).

Suppose Apache has started up, and according to "ps" it's actually running. But when you go to the site, you get:

- `No connection at all`. Make sure that there are no firewalls between you or the server that would filter out packets to the server. Secondly, try using "telnet" to the port you launched the Web server on; for example "telnet myhost.com 80." If you don't get a `Connected to myhost.com` message back, your connection is not even making it to the server in the first place.

- `403 Access Forbidden`. Your document root directory may be unreadable, or you may have something in your access.conf file that prevents access to your site from the machine where your Web browser is.

- `500 Server Error`. Is your front page a CGI script? The script may be failing.

These are the most common errors made in initial server start-ups. If you can establish that contact with the server is actually being made, the next best place to look for error information is in the ErrorLog. Future sections describe each new piece of functionality and also discuss the errors that misconfiguration can bring up.

Apache-SSL

At this point, we will take a slight detour and discuss setting up a variant of the Apache Web server, Apache-SSL, which can conduct secure transactions over the Secure Sockets Layer protocol. SSL is an RSA public-key based encryption protocol developed by Netscape Communications for use in the Netscape Navigator browser and Netscape Web servers.

Until recently, the only options for doing SSL transactions on the World Wide Web has been to use a proprietary server, such as the Netscape Commerce server or the OpenMarket Secure server. Strongly encrypting versions of these servers have not been available outside the United States due to export restrictions in the U.S.

Eric Young, author of the widely used libdes package, along with Tim Hudson, wrote a library that implements SSL, eponymously named SSLeay. The SSLeay package has since expanded to become an all purpose cryptography and certificate handling library, while retaining the same name, "SSLeay."

Ben Laurie, a member of the Apache Group, then took the SSLeay library and interfaced it with the Apache server, making his patches available to people on the Net. Sameer Parekh, of Community ConneXion, Inc., (hereafter referred to as C2) then took Ben Laurie's patches and built a package legal for use within the United States.

Because the RSA technology used by SSL in the United States is covered by patents owned by RSA Data Security, Inc. (RSADSI) (**http://www.rsa.com**), it is not legal to use the SSLeay package "out-of-the-box" within the United States. C2 licensed the RSA technology to make use of the package legal within the United States, using the "RSAREF" package, produced by RSADSI and Consensus Development Corporation (**http://www.consensus.com**).

Due to export restrictions, it is not legal for someone outside the United States to download and install the C2 Apache-SSL package. In fact, we couldn't even put the SSL patches on the CD-ROM included with this book because the book would suddenly have earned the label "munition" and clearance from the U.S. Government would have been required!

Therefore, the installation process for Apache-SSL differs for those within the United States versus those who live outside the U.S. People within the U.S. can simply download the package from C2, at **http://apachessl.c2.org/**, and install it. Outside the United States, people must separately install SSLeay, and then patch Apache with Ben Laurie's patches from his site.

Within the United States, it is legal to use the version of Apache-SSL available for download from C2 for non-commercial purposes only. In order to use Apache-SSL commercially, people must purchase an Apache-SSL Commerce license from C2. After downloading the package from C2, the installation of the server is rather straightforward.

As with the standard Apache, you must first edit the "Configuration" file to reflect the system and any custom modules you may want installed. The lines regarding SSL in the Configuration file should be ignored. The installation process automatically deals with them.

Next, you need to configure RSAREF for your system. The non-commercial distribution of Apache-SSL comes with full RSAREF source code, so you must edit the rsaref/install/makefile to reflect your system. There is usually not much that needs to be edited in this file, except for the C compiler that you need to use. The commercial version, however, does not come with RSAREF source code, but with RSAREF object code for a number of platforms. If you have the commercial release of Apache-SSL, you need to copy the proper version of the RSAREF library from rsaref/install/objs and place it in rsaref/install/rsaref.a.

Finally, to finish the configuration, run "Configure" in the ssl/ directory. Running "Configure" will give you a list of support platforms for SSLeay. Choose one and run "Configure" again in order to configure SSLeay properly for the platform of your choice.

To build, type **make** at the top-level.

Once the server is successfully compiled, you run "make install," in order to install SSLeay and Apache-SSL into /usr/local/ssl and /usr/local/etc/httpd. The installation installs both an SSL version and a plain non-encrypting version of the Apache server in those locations.

Before you can begin using SSL, however, you need to generate your key/certificate pair for use with SSL. The C2 Apache-SSL distribution comes with the program "genkey" you can use

to generate your key and certificate. Run "/usr/local/ssl/bin/genkey httpd." This generates a public/private RSA keypair for use with the server, and puts it in /usr/local/ssl/private/httpd.key. It also generates a PKCS #10 Certificate Signing Request, which you send to the Certificate Authority of your choice (for example, Verisign) along with the proper documentation. The script also generates a "test certificate," so that you can start using the server immediately, without waiting for your Certificate Authority to reply with a signed certificate.

After the key/certificate pair is generated and installed in the proper location, you are ready to start using the server! First, however, familiarize yourself with the SSL-specific configuration directives to Apache-SSL.

```
SSLCertificateFile filename
```

The filename is the location where your server's certificate is stored. It is either relative to /usr/local/ssl/certs, or, if you provide a full pathname, it's the full path to the certificate file. The "SSLCertificateFile" directive is required.

```
SSLCertificateKeyFile filename
```

The SSLCertificateKeyFile is required, unless the file listed in SSLCertificateFile also contains the key. This filename must be relative to /usr/local/ssl/private. It can't be a full pathname.

```
SSLLogFile filename
```

The SSLLogFile is where Apache-SSL logs specific information regarding SSL for each connection, such as the cipher used, and client-authentication information.

```
SSLVerifyClient   0, 1, or 2
```

SSLVerifyClient determines whether or not the server should use X509 client authentication. 0 means no, 1 means it is optional, and 2 means that a client certificate is required.

```
SSLVerifyDepth depth
```

The SSLVerifyDepth is how far along a certificate chain the server should look for a root Certificate Authority when verifying a given client certificate. If you're not using X509 client certificates, a good default value is probably 1.

```
SSLFakeBasicAuth
```

The SSLFakeBasicAuth directive allows you to use X509 client certificates to provide for Basic HTTP/1.0 authentication for accessing various realms of your Web server's document tree.

NOTE

SSLFakeBasicAuth must be used with "SSLVerifyClient 2." If used with any other SSLVerifyClient setting, it is subject to subversion.

After having installed the certificate and key in /usr/local/ssl/certs/httpd.cert and /usr/local/ssl/private/ httpd.key, you can start the server by merely running /usr/local/etc/httpd/start, which starts up the server with some default configuration files, located in conf/httpd.conf and ssl_conf/httpd.conf.

For people outside the United States, the installation process is more involved. You must obtain the SSLeay package (**ftp://ftp.psy.uq.oz.au/pub/Crypto/SSL/**) and then install it according to the directions in the package.

Second, you must obtain Ben's patches to Apache from his site, at **http://www.algroup.co.uk/ApacheSSL/**. The patches must then be installed to your Apache source tree, according to the directions included with the package.

It should be noted that, at publication time, Verisign (**http://www.verisign.com**, a spin-off from RSA, which was the first Certificate Authority for SSL) had just started signing keys generated for Apache-SSL. Other CA's are expected to crop up - Netscape 2.0 comes with half a dozen others defined and waiting to be recognized. In fact, Netscape 2.0 (and hopefully by the time you read this, other browsers) allow for arbitrary CA's to be used, warning the user that a new CA is being used, but still allowing the encrypted conversation to take place.

Table of Contents

03 - Setting Up a Presence on the World Wide Web

05 - Apache Configuration

Chapter 05 - Apache Configuration

By this point you should have a running, minimal web server. In this chapter, you learn about most of the functionality that comes bundled with the server. This chapter is organized as a series of tutorials, so that new users can get up to speed. Toward the end of the chapter, you dive into some experimental Apache modules as well.

By the time you read this chapter, given the rapid pace of development, there will be some significantly new functionality implemented and released. However, the existing functionality is not likely to change much. The Apache Group has had a strong ethic toward backward compatibility.

In this chapter you will learn how to:

- Configure the MIME types of objects on the server
- Use those MIME types to trigger special actions
- Redirect and alias requests for different parts of your site
- Configure directory indexing
- Set up and use server-side includes
- Set up internal imagemap handling
- Use "cookies" to track user sessions
- Set up configurable logging
- Turn on and use content negotiation
- Configure access control based on hostnames and IP numbers, or passwords.
- Configure "virtual" hosts.
- Customize the server's error messages.

In short, this should cover most of the major functionality of Apache 1.0.

Configuration Basics

The "srm.conf" (also known as the "ResourceConfig" file, which is a directive that can be set in httpd.conf) and "access.conf" (also known as the "AccessConfig" file, also a directive in httpd.conf) files are where most of the configuration related to the actual objects on the server takes place. The names are mostly historical - at one point, when the server was still NCSA, the only thing "access.conf" was good for was setting permissions, restrictions, authentication, and so forth. Then, when directory indexing was added, the cry went out for the capability to control certain characteristics on a directory-by-directory basis. The "access.conf" file was the only one that had any kind of structure for that: the pseudo-htmL "<Directory>" container.

With Apache's revamped configuration file parsing routines, most directives can literally appear anywhere. For example, within "<Directory>" containers in access.conf, within "<VirtualHost>" containers in httpd.conf, and so on. However, for sanity's sake, you should keep some structure to the configuration files. You should put server-processing-level configuration options in httpd.conf (like "Port," "<VirtualHost>" containers, etc.), put generic server resource information in "srm.conf" (like "Redirect," "AddType," directory indexing information, etc.), and per-directory configurations in "access.conf."

In addition to the "<Directory>" container, there is the "<Limit>" container, which is used within "<Directory>" containers to specify certain HTTP methods to which particular directives apply. Examples will be given later in this chapter.

Per-Directory Configuration Files

Before you get too deep into the long list of features, take a look at a mechanism that controls most of those features on a directory-by-directory basis by using a file in that directory itself. You can already control subdirectory options in access.conf, as outlined in the previous chapter. However, for a number of reasons, you may want to allow these configurations to be maintained by people other than those who have the power to restart the server (such as people maintaining their home pages), and for that purpose the "AccessFileName" directive was invented.

The default "AccessFileName" is ".htaccess." If you want to use something else, for example ".acc," you would say the following in the srm.conf file:

```
AccessFileName .acc
```

If looking for this file is enabled, and a request comes in that translates to the file /www/htdocs/path/path2/file, the server will look for /.acc, /www/.acc, /www/htdocs/.acc, /www/htdocs/path/.acc, and /www/htdocs/path/path2/.acc, in that order. Also, it will parse the file if it finds it to see what configuration options apply. Remember that this parsing has

to happen with each hit, separately, so this can be a big performance hit. If you turn it off by setting the following in your access config file:

```
<Directory />
AllowOverride None
</Directory>
```

For the sake of brevity and clarity, let's call these files ".htaccess" files. What options can these files affect? The range of available options is controlled by the "AllowOverride" directive within the <Directory> container in the AccessConfig file, as mentioned previously. The exact arguments to "AllowOverride" are as follows:

Argument	Result
AuthConfig	When listed, ".htaccess" can specify their own authentication directives, such as "AuthUserFile," "AuthName," "AuthType," "require," and so on.
FileInfo	When listed, ".htaccess" can override any settings for metainformation about files, using directives such as "AddType," "AddEncoding," "AddLanguage," and so forth.
Indexes	When listed, ".htaccess" files can locally set directives that control the rendering of the directory indexing, as implemented in the module "mod_dir.c." For example, "FancyIndexing," "AddIcon," "AddDescription," and the like.
Limit	Allow the use of the directives that limit access based on hostname or host IP number (allow, deny, and order).
Options	Allow the use of the "Options" directive.
All	Allow all of the above to be true.

< "AllowOverride" options are not merged, which means that if the configuration for "/path/" is different than the configuration for "/", the "/path/" one will take precedence because it's "deeper."

Mime Types: AddType and AddEncoding

A fundamental element of the HTTP protocol, and the reason why the Web was so natural as a home for multiple media formats, is that every data object transferred through HTTP had an associated MIME type. What does this mean?

NOTE

MIME stands for Multipurpose Internet Mail Extensions, and its origins lie in an effort to standardize the transmission of documents of multiple media through e-mail. Part of the MIME specification was that email-messages could contain meta-information in the headers - information **about** the information being sent. One type of MIME header is "Content-Type," which states the format or data type the object is in. For example, htmL is given the label "text/html," JPEG images are given the label "image/jpeg," etc. There is a registry of MIME types maintained by the Internet Assigned Numbers Authority, at

http://www.isi.edu/div7/iana/.

When a browser asks a server for an object, the server gives that object to the browser and states what its "Content-Type" is, and the browser can make an intelligent decision about how to render the document. It can send it to an image program, to a postscript viewer, to a VRML viewer, etc.

What this means to the server maintainer is that every object being served out must have the right MIME type associated with it. Fortunately, there has been a convention of expressing data type through two-, three-, or four-letter suffixes to file name - i.e., "foobar.gif" is most likely to be a gif image.

What the server needs is a file to map the suffix to the MIME content type. Fortunately, Apache comes with such a file in its config directory, a file called mime.types. You'll see that the format of this file is simple. The format consists of one record per line, where a record is a MIME type and a list of acceptable suffixes. This is because while more than one suffix may map to a particular MIME type, you can't have more than one MIME type per suffix. You can use the "TypesConfig" directive to specify an alternative location for the file.

The Internet is evolving so quickly that it would be hard to keep that file completely up to date. To overcome that, you can use a special directive called "AddType," which can be put in an "srm.conf" file like the following:

```
AddType x-world/x-vrml wrl
```

Now, whenever the server is asked to serve a file that ends with ".wrl," it knows to also send a header like the following:

```
Content-type: x-world/x-vrml
```

Thus, you don't have to worry about reconciling future distributions of the "mime.types" file with your private installations and configuration.

As you'll see in future pages, however, "AddType" is also used to specify "special" files that get magically handled by certain features within the server.

A sister to "AddType" is "AddEncoding." Just as the MIME header "Content-Type" can specify the data format of the object, the header "Content-Encoding" specifies the **encoding** of the object. An encoding is an attribute of the object as it is being transferred or stored; semantically, the browser should know that is has to "decode" whatever it gets based upon the listed encoding.The most common use is with compressed files. For example, if you have

```
AddEncoding x-gzip gz
```

and if you then access a file called "myworld.wrl.gz," the MIME headers sent in response will look like the following:

```
Content-Type: x-world/x-vrml
Content-Encoding: x-gzip
```

And any browser worth its two cents will know "Oh, I have to uncompress the file before handing it off to the VRML viewer.

Alias, ScriptAlias, and Redirect

These three directives, all denizens of srm.conf, and all three implemented by the module "mod_alias.c," allow you to have some flexibility with the mapping between "URL-Space" on your server and the actual layout of your file system.

If that last statement sounded cryptic, don't worry. What it basically means is that any URL that looks like "http://myhost.com/x/y/z" does not have to necessarily map to a file named "x/y/z" under the document root of the server:

```
Alias /path/ /some/other/path/
```

The preceding directive will take a request for an object from the mythical subdirectory "/path" under the document root and map it to another directory somewhere else entirely. For example, a request for

```
http://myhost.com/statistics/
```

might normally go to "document root"/statistics, except that for whatever reason you wanted it to point somewhere else outside of the document root.
Say /usr/local/statistics. For that you'd have the following:

```
Alias /statistics/ /usr/local/statistics/
```

To the outside user this would be completely transparent. If you use Alias, it's wise not to alias to somewhere else inside of document root. Furthermore, a request like

```
http://myhost.com/statistics/graph.gif
```

would get translated into a request for the file

```
/usr/local/statistics/graph.gif
```

"ScriptAlias" is just like "Alias," with the side-effect of making everything in the subdirectory by default a CGI script. This might sound a bit bizarre, but the early model for building Web sites had all the CGI functionality separated into a directory by itself, and referenced through the Web server as shown in the following:

```
http://myhost.com/cgi-bin/script
```

If you have in your srm.conf

```
ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-bin/
```

then the preceding URL points to the script at "/usr/local/etc/httpd/cgi-bin/script." As you'll see in a page or two, there is another way to specify that a file is a CGI script to be executed.

"Redirect" does just that - it redirects the request to another resource. That resource could be on the same machine, or somewhere else on the Net. Also, the match will be a substring match, starting from the beginning. For example, if you did:

```
Redirect /newyork http://myhost.com/maps/states/newyork
```

then a request for

```
http://myhost.com/newyork/index.html
```

will get redirected to

```
http://myhost.com/maps/states/newyork/index.html
```

Of course, the second argument to Redirect can be a URL at some other site. Just make sure that you know what you're doing. Also, be wary of creating loops accidentally. For example,

```
Redirect /newyork http://myhost.com/newyork/newyork
```

can have particularly deleterious effects on the server!

A Better Way to Activate CGI Scripts

You read earlier that there is a more elegant way of activating CGI scripts than using "ScriptAlias." You can use the AddType directive and a "magic" MIME type, like so:

```
AddType application/x-httpd-cgi cgi
```

When the server gets a request for a ".cgi" file, it maps to that MIME type, and then catches itself and says "Aha! I need to execute this instead of just dish it out like regular files." Thus, you can have CGI files in the same directories as your htmL and gif and all your other files.

A later chapter will go into more detail about the implementation of CGI in Apache.

Directory Indexing

When Apache is given a URL to a directory, instead of to a particular file, for example

```
http://myhost.com/statistics/
```

Apache first looks for a file specified by the "DirectoryIndex" directive in "srm.conf." In the default configs, this is "index.html." You can set a list of files to search for, or even an absolute path to a page or CGI script:

```
DirectoryIndex index.cgi index.html /cgi-bin/go-away
```

The preceding directive says to look for an "index.cgi" in the directory first. If that can't be found, then look for an "index.html" in the directory. If neither can be found, then redirect the request to "/cgi-bin/go-away."

If it all fails to find a match, then Apache will create, completely on the fly, an htmL listing of all the files available in the directory:

```
<Give a figure here of the directory listing output>
```

There are quite a few ways to customize the output of the directory indexing functionality. First, you need to ask yourself if you care about seeing things like icons, last-modified times, etc., in the reports. If you do, then you want to turn to

```
FancyIndexing On
```

otherwise you'll just get a simple menu of the available files, which you may want for security or performance reasons.

With that going on, you must ask whether you need to customize it further, and how. The default settings for the directory indexing functionality are already pretty elaborate.

The AddIcon, AddIconByEncoding, and AddIconByType directives customize the selection of icons next to files. AddIcon matches icons at the filename level by using the pattern

```
AddIcon iconfile filename [filename] [filename]...
```

Thus, for example,

```
AddIcon /icons/binary.gif .bin .exe
```

means that any file which ends in ".bin" or ".exe" should get the "binary.gif" icon attached. The file names can also be a wildcard expression, a complete file name, or even one of two "special" names: "^^DIRECTORY^^" for directories and "^^BLANKICON^^" for blank lines. So you can see lines like

```
AddIcon /icons/dir.gif ^^DIRECTORY^^
AddIcon /icons/old.gif *~
```

Finally, the "iconfile" can actually also be a string containing both the iconfile's name and the alternate text to put into the ALT attribute. So, your examples should really be

```
AddIcon (BIN,/icons/binary.gif) .bin .exe
```

```
AddIcon (DIR,/icons/dir.gif) ^^DIRECTORY^^
```

The "AddIconByType" directive is actually a little bit more flexible and probably comes more highly recommended in terms of actual use. Instead of tying icons to file name patterns, it ties icons to the MIME type associated with the files. The syntax is very roughly the same:

```
AddIconByType iconfile mime-type [mime-type]...
```

"Mime-type" can be either the exact MIME type matching what you have assigned a file, or it can be a pattern match. Thus, you see entries in the default configuration files like the following:

```
AddIconByType (SND,/icons/sound2,gif) audio/*
```

This is a lot more robust than trying to match against file name suffixes.

"AddIconByEncoding" is used mostly to distinguish compressed files from the others. This makes sense only if used in conjunction with "AddEncoding" directives in your "srm.conf" file. The default "srm.conf" has these entries:

```
AddEncoding x-gzip gz
AddEncoding x-compress Z
AddIconByEncoding (CMP,/icons/compressed.gif)
    x-compress x-gzip
```

This will set the icon next to compressed files appropriately.

The "DefaultIcon" directive specifies the icon to use when none of the patterns match a given file when the directory index is generated.

```
DefaultIcon /icons/unknown.gif
```

It is possible to add text to the top and the bottom of the directory index listing. This capability is very useful as it turns the directory indexing capabilities from just a UNIX-like interface into a real dynamic document interface. There are two directives to control this: "HeaderName" and "ReadmeName," which specify the file names for the content at the top and bottom of the listing, respectively. Thus, as shown in the default srm.conf file:

```
HeaderName HEADER
ReadmeName README
```

When the directory index is being built, Apache will look for "HEADER.html." If it finds it, it'll throw the content into the top of the directory index. If it fails to find that file, it'll look for just "HEADER," and if it finds that it will presume the file is plain text and do things like escape characters such as "<" to "<", and then insert it into the top of the directory index. The same process happens for the file "README," except that the resulting text goes into the bottom of the generated directory index.

In many cases, be it for consistency or just plain old security reasons, you will want to have the directory indexing engine just ignore certain types of files, like Emacs backup files or files beginning with a ".". The "IndexIgnore" directive addresses this; the default setting is

```
IndexIgnore */.??* *~ *# */HEADER* */README* */RCS
```

This line might look cryptic, but it's basically a space-separated list of patterns. The first pattern matches against any "." file that is longer than 3 characters. This is so that the link to the higher-up directory ("..") can still work. The second ("*~") and third ("*#") are common patterns for matching old emacs backup files. The next ones are to avoid listing the same files used for "HeaderName" and "ReadmeName" as in the preceding. The last ("*/RCS") is given because many sites out there use "RCS," a software package for revision control maintenance, which stores its extra (rather sensitive) information in "RCS" directories.

Finally you get to two really interesting directives for controlling the last set of options regarding directory indexing. The first is "AddDescription," which works similarly to "AddIcon."

```
AddDescription description filename [filename]...
```

I.e.

```
AddDescription "My cat" /private/cat.gif
```

As elsewhere, "file name" can actually be a pattern, so you can have

```
AddDescription "An MPEG Movie Just For You!" *.mpg
```

Finally, you have the granddaddy of all options-setting directives, "IndexOptions." This is the smorgasbord of functionality control. The syntax is simple:

```
IndexOptions option [option]...
```

And the list of available options are as follows:

Option	Explanation
FancyIndexing	This is the same as the separate "FancyIndexing" directive. Sorry to confuse everyone, but backward compatibility demands bizarre things sometimes!
IconsAreLinks	If this is set the icon will be clickable as a link to whatever resource the entry it is associated

	with links to. In other words, the icon becomes part of the hyperlink.
ScanhtmLTitles	When given a listing for an htmL file, the server will open the htmL file and parse it to obtain the value of the <TITLE> field in the htmL document, if it exists. This can put a pretty heavy load on the server, since it's a lot of disk accessing and some amount of CPU to extract the title from the htmL, so it's not recommended unless you know you have the capacity.
SuppressDescription, SuppressLastModified, SuppressSize	These will suppress their respective fields in the directory indexing output. Normally each of those (Description, Last Modified, Size) is a field in the output listings.

By default none of these are turned on. The options do not "merge," which means that when you are setting these on a per-directory basis by using either access.conf or .htaccess files, setting the options for a more specific directory requires resetting the complete options listing. For example, envision the following in your access configuration file:

```
<Directory /pub/docs/>
IndexOptions ScanhtmLTitles
</Directory>
<Directory /pub/docs/others/>
IndexOptions IconsAreLinks
</Directory>
```

Directory listings done in or below the second directory, "/pub/docs/others/,", would not have "ScanhtmLTitles" set. Why? Well, you figured administrators would need to be able to disable an option they had set globally in a specific directory, and this was simpler than writing "NOT" logic into the options listings.

If you run into problems getting directory indexing to work, make sure that the settings you have for the "Options" directive in the access config files allow for directory indexing in that directory. Specifically, the "Options" directive must include "Indexing." Furthermore, if you are using .htaccess files to set things like "AddDescription" or "AddIcon," the "AllowOverride" directive must include in its list of options "FileInfo." This is covered in more depth later on in this chapter.

User Directories

Sites with many users sometimes prefer to be able to give their users access to managing their own parts of the Web tree in their own directories, using the

URL semantics of

```
http://myhost.com/~user/
```

Where "~user" is actually an alias to a directory in the user's home directory. This is different from the "Alias" directive, which could only map a particular pseudo-directory into an actual directory. In this case, you want "~user" to map to something like "/home/user/public_html," and because the number of "users" can be very high, some sort of macro is useful here. That macro is the directive "UserDir."

With "UserDir" you specify the subdirectory within the users' home directory where they can put content, which is mapped to the "~user" URL. So in other words, the default

```
UserDir public_html
```

will cause a request for

```
http://myhost.com/~eric/index.html
```

to cause a lookup for the UNIX file

```
/home/eric/public_html/index.html
```

presuming that "/home/eric" is eric's home directory. The default of "public_html" is a historical artifact more than anything else. There's no reason why you can't make it "Web_stuff" or something like that.

Apache 1.1's user-directory module will have even more functionality, but at press time the feature set has not been nailed down.

Special Modules

Most of the functionality that distinguishes Apache from the competition has been implemented as modules to the Apache API. This has been extremely useful in allowing functionality to evolve separately from the rest of the server, and for allowing for performance tuning. This section will cover that extra functionality in detail.

Server Side Includes

Server side includes are best described as a preprocessing language for htmL. The "processing" takes place on the server side, such that visitors to your site never need know that you use server side includes, and thus requires no special client software. The format of these includes looks something like the following:

```
<!--#directive attribute="value" -->
```

Sometimes a given "directive" can have more than one attribute at the same time. The funky syntax is due to the desire to hide this functionality within an SGML comment - that way your regular htmL validation tools will work without having to learn new tags or anything. The syntax is important; leaving off the final "--," for example, will result in errors.

#include

This directive is probably the most commonly used directive. It is used to insert another file into the htmL document. The allowed attributes for this directive are "virtual" and "file." The functionality of the "file" attribute is a subset of that provided by the "virtual" attribute, and it exists mostly for backward compatibility, so its use is not recommended.

The "virtual" attribute instructs the server to treat the value of the attribute as a request for a relative link - meaning that you can use "../" to locate objects above the directory, and that other transforms like Alias will apply.

For example:

```
<!--#include virtual="quote.txt" -->
<!--#include virtual="/toolbar/footer.html" -->
<!--#include virtual="../footer.html" -->
```

#exec

This directive is used to run a script on the server side and insert its output into the SSI document being processed. There are two choices: executing a CGI script by using the "cgi" attribute, or executing a shell command by using the "cmd" attribute.

For example:

```
<!--#exec cgi="counter.cgi" -->
```

would take the output of the CGI program "counter.cgi" and insert it into the document. Note that the CGI output still has to include the "text/html" content type header or an error will occur.

Likewise,

```
<!--#exec cmd="ls -l" -->
```

would take the output of a call to "ls -l" in the document's directory and insert it. Like the "file" attribute to the #include directive, this is mostly for backward compatibility, because it is something of a security hole in an untrusted environment.

NOTE

There are definitely security concerns with allowing users access to CGI functionality, and even greater concerns with "#exec cmd", such as cmd="cat /etc/passwd". If the site administrator wishes to allow people to use server-side includes, but not to use the #exec directive, then they can set "IncludesNOEXEC" as an option for the directory in the access configurations.

#echo

This directive has one attribute, "var," whose value is any CGI environment variable as well as a small list of other variables:

Attribute	Defintion
DATE_GMT	The current date in Greenwich Mean Time.
DATE_LOCAL	The current date in the local time zone.
DOCUMENT_NAME	The file system name of the SSI document, not including the directories below it.
DOCUMENT_URI	In a URL of the format "http://host/path/file." This is the "/path/file" part.
LAST_MODIFIED	The date the SSI document was modified.

Example:

```
<!--#echo var="DATE_LOCAL" -->
```

This will insert something like "Wednesday, 06-Mar-96 10:44:54 GMT" into the document.

#fsize, #flastmod

These two directives print out the size and the last-modified date, respectively, of any object given by the URI listed in the "file" or "virtual" attribute, as in the #include directive. For example

```
<!--#fsize file="index.html" -->
```

would return the size of the index.html file in that directory.

#config

You can modify the rendering of certain SSI directives by using this directive.

The "sizefmt" attribute controls the rendering of the "#fsize" directive with values of "bytes" or "abbrev." The exact number of bytes is printed when "bytes" is given, whereas an abbreviated version of the size (either in K for kilobytes or M for megabytes) is given when "abbrev" is set.

Thus, for example, a snippet of SSI htmL like

```
<!--#config sizefmt="bytes" -->
The index.html file is <!--#fsize
  virtual="index.html" --> bytes
```

would return "The index.html file is 4,522 bytes." Meanwhile, if

```
<!--#config sizefmt="abbrev" -->
```

was used, "The index.html file is 4K bytes" would be returned. The default is "abbrev."

The "timefmt" directive controls the rendering of the date in the "DATE_LOCAL," "DATE_GMT," and "LAST_MODIFIED" values for the #echo directive. It uses the same format as the "strftime" call (In fact, that's what the server does. It calls "strftime.") This format consists of variables that begin with "%." For example, "%H" is the hour of the day, in 24-hour format. The list of variables is best found by consulting your system's "man" page by typing **man strftime** for directions as to how to construct a "strftime"-format date string.

An example, though, might be:

```
<!--#config timefmt="%Y/%m/%d-%H:%M:%S" -->
```

and the resulting date string for Jan. 2nd, 1996, at 12:30 in the afternoon would thus be

```
1996/01/02-12:30:00
```

Finally, the last attribute the "config" directive can take is "errmsg," which is simply the error to print out if there are any problems parsing the document. For example, the right default is:

```
<!--#config errmsg="An error occurred while
processing this directive" -->
```

Internal Imagemap Capabilities

The default imagemap module supplied with Apache allows you to reference imagemaps without using or needing any CGI programs. This functionality is contained in the "mod_imap" module. First, you add to your srm.conf yet another magic AddType directive:

```
AddType application/x-httpd-imap map
```

This now means that any file ending with ".map" will be recognized as an imagemap file. After restarting the server to pick up the change, one can make reference to a .map file directly.

Look at an example: the following document, index.html, has an imagemap on it, where the image is "usa.jpg" and the mapfile is "usa-map.map." The htmL to build that imagemap would look like:

```
<A HREF="usa-map.map"><IMG SRC="usa.jpg" ISMAP></A>
```

Imagemaps are covered in more detail in a later chapter--the only important thing from a configuration standpoint is that the magic content type is activated.

Cookies

HTTP "cookies" are a method for maintaining statefulness in a stateless protocol. What does this mean? In HTTP, a session between a client and a server typically spans many separate actual TCP connections, thus making it difficult to tie together accesses into an application that requires state, such as a shopping cart application. Cookies are a solution to that problem. As implemented by Netscape in their browser and subsequently by many others, servers can assign clients a "cookie," meaning some sort of opaque string whose meaning is significant only to the server itself, and then the client can give that cookie back to the server on subsequent requests.

The module "mod_cookies" nicely handles the details of assigning unique cookies to every visitor, based on their hostname and a random number. This cookie can be accessed from the CGI environment as the HTTP_COOKIE environment variable, for the same reason that all HTTP headers are accessible to CGI applications. The CGI scripts can use this as a key in a session tracking database, or it can be logged and tallied up to get a good, if undercounted, estimate of the total number of users that visited a site, not just the number of hits or even number of unique domains.

Happily, there are no configuration issues here - simply compile with mod_cookies and away you go. Couldn't be easier.

Configurable Logging

For most folks, the default logfile format (also known as Common Logfile Format, or CLF) does not provide enough information when it comes to doing a serious analysis of the efficacy of your web site. It provided basic numbers in terms of raw hits, pages accessed, hosts accessing, timestamps, etc., but it fails to capture the "referring" URL, the browser being used, and any cookies being used as well. So, there are two ways to get more data for your logfiles: by using the NCSA-compatibility directives for logging certain bits of info to separate browsers, or using Apache's own totally configurable logfile format.

NCSA Compatibility

For compatibility with the NCSA 1.4 Web server, two modules were added. These modules log the "User-Agent" and "Referer" headers from the HTTP request stream.

"User-Agent" is the header most browsers send that identify what software the browser is using. Logging of this header can be activated by an "AgentLog" directive in the srm.conf file, or in a virtualhost-specific section. This directive takes one argument, the name of the file to which the user-agents are logged. For example:

```
AgentLog logs/agent_log
```

To use this, though, you need to ensure that the "mod_log_agent" module has been compiled and linked to the server.

Similarly, the "Referer" header is sent by the browser to indicate the tail end of a link - in other words, when you are on a page with a URL of "A," and there is a link on that page with a URL of "B," and you follow that link, the request for page "B" includes a "Referer" header with the URL of "A." This is very useful for finding what sites out there link to your site, and what proportion of traffic they account for.

The logging of this header is activated by a "RefererLog" directive, which points to the file to which the referers get logged.

```
RefererLog logs/referer_log
```

One other option the Referer logging module provides is RefererIgnore, a directive that allows you to ignore "Referer" headers, which contain some string. This is useful for weeding out the Referers from your own site, if all you are interested in is links to you from other sites. For example, if your site is "www.myhost.com," you might want to use the following:

```
RefererIgnore www.myhost.com
```

Remember that logging of the "Referer" header requires compiling and linking in mod_log_referer.

Totally Configurable Logging

The previous modules were provided, like many Apache features, for backward compatibility. They have some problems, though. Because they don't contain any other information about the request they are logging from, it's nearly impossible to tell which "Referer" fields went to which specific objects on your site. Ideally all the information about a transaction with the server can be logged into one file, extending the "common logfile format" or replacing it altogether. Well, such a beast exists, in the "mod_log_config" module.

This module implements the "LogFormat" directive, which takes as its argument a string, with variables beginning with "%" to indicate different pieces of data from the request. The variables are:

Variable	Definition
%h	Remote host
%l	Remote "identd" identification
%u	Remote user, as determined by any user authentication that may take place. Note that if the user was not authenticated, and the status of the request is a 401, this field may be bogus.
%t	The common logfile format for time.
%r	First line of request
%s	Status. For requests that got internally redirected, this is status of the original request; %>s will give the last.
%b	Bytes sent.
% {Foobar}i	The contents of Foobar: header line(s) in the request from the client to the server.
% {Foobar} o	The contents of Foobar: header line(s) in the response from the server to the client.

So, for example, if you wanted to capture in your log just the remote hostname, the object they requested, and the timestamp, you would do the following:

```
LogFormat "%h \"%r\" %t"
```

And that would log things that looked like

```
 host.outsider.com "GET / HTTP/1.0"
[06/Mar/1996:10:15:17]
```

Note that you really have to use a quote around the request variable - the configurable logging module does not escape the values of the variables. But use a slash-quote, "\"", to distinguish that from the end of the string.

Say you want to add logging of the "User-Agent" string to that as well - in this case, your log format would become:

```
LogFormat "%h \"%r\" %t \"%{User-Agent}i\""
```

Because the User-Agent field typically has spaces in it, it too should be quoted. Say you want to capture the Referer field:

```
LogFormat "%h \"%r\" %t %{Referer}i"
```

You don't need the escaping quotes because Referer headers, since they are URL's, don't have spaces in them. However, if you are building a mission-critical application you might as well quote it as well, because the "Referer" header is supplied by the client and thus there are no guarantees about its format.

The default format is the Common Logfile Format (CLF), which in this syntax is expressed as

```
LogFormat "%h %l %u %t \"%r\" %s %b"
```

In fact, most existing logfile analysis tools for CLF will ignore extra fields tacked onto the end, so to capture the most important extra information and yet still be parseable by those tools, you might want to use the format:

```
LogFormat "%h %l %u %t \"%r\" %s %b
%{Referer}i \"%{User-Agent}i\""
```

Power users take note: If you want even more control over what gets logged, you can use the configurable logging module to implement a simple conditional test for variables. This way, you can configure it to only log variables when a particular status code is returned, or not returned. The format for this is to insert a comma-separate list of those codes between the "%" and the letter of the variable, like so:

```
%404,403{Referer}i
```

This means that the Referer header will only be logged if the status returned by the server is a "404 Not Found," or a "403 Access Denied." All other times just a "-" is logged. This would be useful if all you cared about using "Referer" for was to find out old links that point to resources no longer available.

The negation of that conditional is to put a "!" at the beginning of the list of status codes; so for example,

```
%!401u
```

will log the user in any user authentication transaction, unless the authentication failed, in which case you probably don't want to see the name of the bogus user anyway.

Remember that, like many functions, this can be configured per virtual host. Thus, if you want all logs from all virtual hosts on the same server to go to the same log, you might want to do something like

```
LogFormat "hosta ...."
```

in the <virtualhost> sections for "hosta" and

```
LogFormat "hostb ...."
```

in the <virtualhost> sections for "hostb." More details about virtual hosts will appear later in this chapter.

A key note: You have to compile in "mod_log_config" for this functionality. You must also make sure that the default logging module, "mod_log_common," is not compiled in, or the server will get confused.

Content Negotiation

Content negotiation is the mechanism by which a Web client can express to the server what data types it knows how to render, and based on that information, the server can give the client the "optimal" version of the resource requested. Content negotiation can happen on a number of different characteristics - the content type of the data (also called the "media type"), the human language the data is in (English? French? etc.), the character set of the document, and its encodings.

Content Type Negotiation

For example, say you want to use inlined JPEG images on your pages. You don't want to alienate people using older browsers, which don't know how to inline JPEG images, so you also make a gif version of that image. Even though the gif might be larger or only 8-bit, that's still better than giving the browser something it can't handle, causing a broken link. So, the browser and the server "negotiate" for which data format the server sends to the client.

The specifications for content negotiation have been a part of HTTP since the beginning. Unfortunately, it can not be relied upon as extensively as one would like. For example, current browsers that implement "plug-ins," by and large do not express in the connection headers which media types they have plug-ins for. Thus, content-negotiation can't be used to decide whether to send someone a "ShockWave" file or its Java equivalent, currently. The only safe place to use it currently is to distinguish between inlined JPEG or gif images on a page. Enough browsers in use today implement content negotiation closely enough to get this functionality.

The mod_negotiation.c in Apache 1.0 implements the content negotiation specifications in an older version of the HTTP/1.0 IETF draft, which at the time of this writing is on its way to informational RFC status. It was removed because the specification was not entirely complete, and a document describing it could not be labeled "Best Current Practice," which is what the HTTP/1.0 specification became. Content negotiation is getting significantly enhanced for HTTP/1.1. However, this doesn't mean it can't be safely used now for inlined image selection.

To activate it, you must include the module "mod_negotiation.c" into the server. There are actually two ways to configure content negotiation:

- Using a "type-map" file describing all the variants of a negotiable resource with specific preference values and content characteristics.
- Setting an "Options" value called "MultiViews".

Since your focus is pragmatic, you will go only into the "MultiViews" functionality. If you are interested in the type-map functionality, the Apache Web site has documentation on it.

In your access.conf file, find the line that sets the "Options" for the part of the site you wish to enable content negotiation within. This may be the whole site, but that's fine. If "MultiViews" is not present in that line, it must be. The "All" value does not, ironically enough, include "MultiViews." This is again for backward compatibility. So, you might have a line that looks like:

```
Options Indexes Includes Multiviews
```

or

```
Options All MultiViews
```

Once this change is made, restart your server to pick up the new configuration.

With this turned on, you can do the following: place a JPEG image in a directory, say "/path/," and call it "image.jpg." Now, make an equivalent gif format image, and place it in the same directory, as "image.gif. The URL's for these two objects are

```
http://host/path/image.jpg
```

and

```
http://host/path/image.gif
```

respectively. Now, if you ask your Web browser to fetch,

```
http://host/path/image
```

the server will go into the "/path/" directory, see the two "image" files, and then determine which one to send based on what the client states it can support. In the case where the client says it can accept either JPEG images or gif images equally, the server will choose the version that is the smallest, and send that to the client. Usually, JPEG images are much smaller than gif images.

So, if you made your htmL look something like the following:

```
<htmL><HEAD>
```

```
<TITLE>Welcome to the Gizmo Home Page!</TITLE>
</HEAD><BODY>
<IMG SRC="/header" ALT="GIZMO Logo">
Welcome to Gizmo!
<IMG SRC="/products" ALT="Products">
<IMG SRC="/services" ALT="Services">
```

then you can have separate gif and JPEG files for "header" "products," and "services," and the clients will for the most part get what they claim they can support.

Note that, if you have a file called "image" and a file called "image.gif," the file called "image" will be requested no matter if a request is made for just "image." Likewise, a request specifically for "image.gif" would never return "image.jpg" even if the client knew how to render JPEG images.

Human Language Negotiation

If "MultiViews" is enabled, you can also distinguish resources by the language they are in, such as French, English, Japanese, etc. This is done by adding more entries to the file suffix namespace that map to the languages the server wishes to use, and then giving them a ranking that ties can be broken. Specifically, in the "srm.conf" file, go two new directives, "AddLanguage" and "LanguagePriority." The formats are as follows:

```
AddLanguage en .en
AddLanguage it .it
AddLanguage fr .fr
AddLanguage jp .jp
LanguagePriority en fr jp it
```

Say you want to use this to negotiate on the file "index.html," which you had available in English, French, Italian, and Japanese. You would create an "index.html.en," "index.html.fr," "index.html.it," and "index.html.jp," respectively, and then reference the document as "index.html." When a multilingual client connects, it should indicate in one of the request headers ("Accept-Language," to be specific) which languages it prefers, and it expresses that in standard two-letter notation. The server sees what the clients can accept, and gives them "the best one." LanguagePriority is what organizes that decision of "the best one." If English is unacceptible to the client, try French, otherwise try Japanese, otherwise try Italian. LanguagePriority also states which one should be served if there is no "Accept-Language" header.

Because the language mapping suffixes and the content-type suffixes share the same namespace, you can mix them around. "index.fr.html" is the same as "index.html.f.," Just make sure that you reference it with the correct negotiable resource.

As-Is Files

Often, you might like to request specific HTTP headers in your documents, such

as Expires:, but you don't want to make the page a CGI script. The easiest way is to use the "httpd/send-as-is" magic MIME type.

```
AddType httpd/send-as-is asis
```

This means that any file that ends in ".asis" can include its own MIME headers. However, it **must** include **two** carriage returns before the actual body of the content. Actually, it should include two carriage return / line feed combinations, but Apache is forgiving and will insert that for you. So, if you wanted to send a document with a special unique custom MIME type you didn't want registered with the server, you can send:

```
Content-type: text/foobar

This is text in a very special "foobar" MIME type.
```

The most significant application I've run across for this is as an extremely efficient mechanism for doing server-push objects without CGI scripts. The reason a CGI script is needed to create a server-push usually is that the Content-type usually includes the multipart separator (since a server-push is actually a MIME multipart message). I.e.:

```
Content-type: multipart/x-mixed-replace;boundary=XXXXXXXX

--XXXXXXXX
Content-type: image/gif

....(gif data)....
--XXXXXXXX
Content-type: image/gif

....(gif data)....
--XXXXXXXX
....
```

By making this stream of data a simple file instead of a CGI script, you save yourself potentially a lot of overhead. Just about the only thing you lose is the ability to do timed pushes, but for many people their slow internet connection acts as a sufficient time valve already.

It should also be pointed out that if you have MultiViews turned on, you can add an ".asis" to the end of a file name and none of your links need to be renamed. I.e. "foobar.html" can easily become "foobar.html.asis," while still being able to call it "foobar.html."

One last compelling application of "asis" is being able to do HTTP redirection without needing access to server config files. For example, the following .asis file will redirect people to another location:

```
Status 302 Moved
Location: http://some.other.place.com/path/
Content-type: text/html
```

```
<htmL>
<HEAD><TITLE>We've Moved!</TITLE></HEAD>
<BODY>
<H1>We used to be here, but now we're
<A HREF="http://some.other.place.com/path/">over there. </A>
</H1>
</BODY></htmL>
```

The htmL body is there simply for clients which don't understand the 302 response.

Advanced Functionality

Host-Based Access Control

One can control access to the server, or even a subdirectory of the server, based on the hostname, domain, or IP number of the client's machine. This is done by using the directives "allow" and "deny," which can be used together at the same time by using "order." "allow," and "deny" can take multiple hosts:

```
deny from badguys.com otherbadguys.com
```

Typically, you want to do one of two things: you want to deny access to your server from everyone but a few other machines, or you want to grant access to everyone except a few hosts. The first case is handled as follows:

```
order allow,deny
allow from mydomain.com
deny from all
```

This means, "only grant access to hosts in the domain 'mydomain.com'." This could include "host1.mydomain.com," "ppp.mydomain.com," "the-boss.mydomain.com," etc.

The "order" directive above tells the server to evaluate the "allow" conditions before the "deny" conditions when determining whether to grant access. Likewise, the "only exclude a couple of sites" case described above can be handled by using:

```
order deny,allow
deny from badguys.com
allow from all
```

"order" is needed because, again mostly for historical reasons, the order in which directives appear is not significant. Thus, the server needs to know which rule to apply first. The default for "order" is "deny,allow."

There is a third argument to "order," called "mutual-failure," in which a condition has to pass both the "allow" and "deny" rules in order to succeed. In other words, it has to appear on the "allow" list, and it must not appear on the "deny" list. For example,

```
order mutual-failure
allow from mydomain.com
deny from the-boss.mydomain.com
```

In this example, "the-boss.mydomain.com" is prevented from accessing this resource, but every other machine at "mydomain.com" can access it.

It should be mentioned at this point that protecting resources by hostname is dangerous. It is relatively easy for a determined persons who control the reverse-DNS mapping for their IP number to spoof any hostname they want. Thus, it is strongly recommended that you use IP numbers to protect anything sensitive. In the same way you can simply list the domain to refer to any machine in that domain, you can also give fragments of IP numbers:

```
allow from 204.62.129
```

This will only allow hosts whose IP numbers match that, such as "204.62.129.1" or "204.62.129.130."

Typically these directives are used within a "<Limit>" container, and even that within a "<Directory>" container, usually in an access.conf configuration file. The following example is a good template for most protections; it protects the directory "/www/htdocs/private" from any host except those in the "204.62.129" IP space.

```
<Directory /www/htdocs/private>
Options Includes
AllowOverride None
<Limit GET POST>
order allow,deny
deny from all
allow from 204.62.129
</Limit>
</Directory>
```

User Authentication

When you place a resource under "user authentication," you restrict access to it by requiring a name and password. This name and password is kept in a database on the server. This database can take many forms, and in fact Apache modules have been written to access flat file databases, DBM file databases, Msql databases (a freeware database), Oracle and Sybase databases, and more. This book covers only the flat-file and DBM-format databases.

First, some basic configuration directives. The "AuthName" directive sets the authentication "Realm" for the password-protected pages. The "Realm" is what gets presented to clients when prompted for authentication - "Please enter your name and password for the realm ."

The "AuthType" directive sets the "authentication type" for the area. In HTTP/1.0 there is only one authentication type, and that is "Basic." HTTP/1.1

will have a few more, such as "MD5."

The "AuthUserFile" directive specifies the file thT contains a list of names and passwords, one pair per line, where the passwords are encrypted by using the simple Unix crypt() routines. I.e.

```
joe:D.W2yvlfjaJoo
mark:21slfoUYGksIe
```

The "AuthGroupFile" directive specifies the file which contains a list of groups, and members of those groups, separated by spaces. For example:

```
managers: joe mark
production: mark shelley paul
```

Finally, the "require" directive specifies what conditions need to be met for access to be granted. It can list only a specified list of users who may connect, it can specify a group or list of groups of users who may connect, or it can say any valid user in the database is automatically granted access. For example:

```
require user mark paul
(Only mark and paul may access.)

require group managers
(Only people in group managers may access.)

require valid-user
(Anyone in the AuthUserFile database may access.)
```

The configuration file ends up looking something like this:

```
<Directory /www/htdocs/protected/>
AuthName Protected
AuthType basic
AuthUserFile /usr/local/etc/httpd/conf/users
<Limit GET POST>
require valid-user
</Limit>
</Directory>
```

If you want to protect it to a particular group, the configuration file looks something like the following:

```
<Directory /www/htdocs/protected/>
AuthName Protected
AuthType basic
AuthUserFile /usr/local/etc/httpd/conf/users
AuthGroupFile /usr/local/etc/httpd/conf/group
<Limit GET POST>
require group managers
</Limit>
</Directory>
```

DBM Authentication

Apache can be configured to also use DBM files for faster password and group-membership lookups. To use this, you must have the "mod_auth_dbm" module compiled into the server.

"DBM" files are UNIX file types that implement a fast hashtable lookup, making them ideal for handling large user/password databases. The flat-file systems requires parsing the password file for every access until a match is found, potentially going through the entire file before returning a `can't find that user` error. Hash tables, on the other hand, know instantly whether a "key" exists in the database, and what its value is.

Different systems implement them slightly differently - some use the "ndbm" libraries, some use the "berkeley db" libraries, but the interface through Apache is exactly the same.

To use a DBM file for the database instead of a regular flat file, you use a different directive, "AuthDBMUserFile" instead of "AuthUserFile." Likewise for the group file - "AuthDBMGroupFile" instead of "AuthGroupFile" is used.

Take a look at creating the DBM files. There is a program supplied in the "support" subdirectory of the Web site called "dbmmanage." It is a file for creating and managing DBM files. The basic syntax is as follows:

```
dbmmanage dbmfile command key [value]
```

"Command" can be one of: add, adduser, view, delete

So, to add a "value" to a DBM file called "users," one would say:

```
dbmmanage users add joe joespassword
```

You have just added a record to the DBM file, with "joe" as the key and "joespassword" as the value. To see this you say:

```
dbmmanage users view joe
```

or if you want to see the whole database,

```
dbmmanage users view
```

However, you want to store encrypted passwords, because that's what the server uses for authentication. For that you use the "adduser" command:

```
dbmmanage users adduser joe joespassword
```

Now, if you do a "view" to look at it, "joespassword" will be replaced by a lot of what looks like junk. Don't worry, that's the encrypted password.

Groups are done a little bit differently in DBM files. Instead of making the key of the database the group, the key is the user and the value is a comma-separated

list of the groups that user is in. For example:

```
dbmmanage group adduser joe managers,production
```

Wait, you say, there's no file called "users." Why do I see a "users.pag" and "users.dir"?

Well, DBM files are pretty weird. They aren't like regular files; they can't be looked at. Some systems implement the hash table by keeping the index separate from the data, as in this example with the ".pag" and ".dir" files. On BSD systems, where Berkeley DB is implemented, DBM files are saved with a ".db" appendix. So, one should get used to the idea that the "name" of a DBM file is actually its filename without the suffix.

The configuration file snippet now looks something like:

```
<Directory /www/htdocs/protected/>
AuthName Protected
AuthType basic
AuthDBMUserFile /usr/local/etc/httpd/conf/users
AuthDBMGroupFile /usr/local/etc/httpd/conf/group
<Limit GET POST>
require group managers
</Limit>
</Directory>
```

Note that "users" and "groups" must be the "name" of the DBM file, as described in the preceding, and pointing to

```
AuthDBMUserFile /usr/local/etc/httpd/conf/users.db
```

would not work.

Make sure that you don't put the user and group databases in the public Web tree, **ever**. Several Web search engines out there have proven themselves to be efficient sources for "/etc/passwd" files unintentionally put on the site. Don't take that risk.

Virtual Hosts

Apache implements a very clean way of handling "virtual hosts," which is the name for the mechanism for being able to serve more than one "host" on a particular machine. Due to a limitation in HTTP, this is accomplished currently by assigning more than one IP number to a machine, and then having Apache bind differently to those different IP numbers. For example, a Unix box might have 204.122.133.1, 204.122.133.2, and 204.122.133.3 pointing to it, with www.host1.com bound to the first, www.host2.com bound to the second, and

www.host3.com bound to the third.

This book will not go into how to configure additional IP addresses for your machine, since that varies completely from platform to platform. Your user manual for the operating system should contain information about configuring additional numbers - this is a standard capability on just about all systems these days.

"Virtual hosts" are configured using a container in httpd.conf. They look something like this:

```
<VirtualHost www.host1.com>
DocumentRoot /www/htdocs/host1/
TransferLog logs/access.host1
ErrorLog logs/error.host1
</VirtualHost>
```

The attribute in the VirtualHost tag is the hostname, which the server looks up to get an IP address. Note that if there is any chance that "www.host1.com" can return more than one number, or if the web server might have trouble resolving that to an IP number at any point, you might want to use the IP number instead.

Any directives put within the VirtualHost container pertain only to requests made to that hostname. The DocumentRoot points to a directory which (presumably) contains content specifically for www.host1.com.

Each virtual host can have its own access log, its own error log, its own derivative of the other logs out there, its own Redirect and Alias directives, its own ServerName and ServerAdmin directives, and more. In fact the only things it can not support, out of the core set of directives, are:

```
ServerType, UserId, GroupId, StartServers, MaxSpareServers,
MinSpareServers, MaxRequestsPerChild, BindAddress, PidFile,
TypesConfig, and ServerRoot.
```

If you plan on running Apache with a large number of virtual hosts, you need to be careful to watch the process limits; for example, some unix platforms only allow processes to open 64 file descriptors at once. An Apache child will consume one file descriptor per logfile per virtual host, so 32 virtualhosts each with their own transfer and error log would quickly cross that limit. You will notice if you are running into problems of this kind if your error logs start reporting errors like "unable to fork()", or your access logs aren't getting written to at all. Apache does try and call setrlimit() to handle this problem on its own, but the system sometimes prevents it from doing so successfully.

Customized Error Messages

Apache can give customized responses in the event of an error. This is controlled using the "ErrorDocument" directive. The syntax is:

```
ErrorDocument <HTTP response code> <action>
```

Where "HTTP response code" is the event which triggers the "action." The "action" can be:

- A local URI to which the server is internally redirected.
- An external URL to which the client is redirected.
- A text string, which starts with a "", and where the %s variable contains any extra information if available.

For example:

```
ErrorDocument 500 "Ack! We have a problem here: %s.
ErrorDocument 500 /errors/500.cgi
ErrorDocument 500 http://backup.myhost.com/
ErrorDocument 401 /subscribe.html
ErrorDocument 404 /debug/record-broken-links.cgi
```

Two extra CGI variables will be passed to any redirected resource: "REDIRECT_URL" will contain the original URL requested, and "REDIRECT_STATUS" will give the original status that caused the redirection. This will help the script if its job is to try and figure out what caused the error response.

Assorted "httpd.conf" Settings

There are a couple last configuration options that fell through the cracks.

BindAddress

At startup, Apache will bind to the port it is specified to bind to, for all IP numbers which the box has available. The "BindAddress" directive can be used to specify only a specific IP address to bind to. Using this, one can run multiple copies of Apache, each serving different virtual hosts, instead of having one daemon which can handle all virtual hosts. This is useful if you want to run two web servers with different system user-id's for security and access control reasons.

For example, let's say you have three IP addresses (1.1.1.1, 1.1.1.2, and 1.1.1.3, with 1.1.1.1 being the primary address for the machine), and you want to run three web servers, yet you want one of them to run as a different user ID than the other two. One would have two sets of configuration files; one would say something like

```
User web3
BindAddress 1.1.1.3
ServerName www.company3.com
DocumentRoot /www/company3/
```

And the other would have

```
User web1
ServerName www.company1.com
DocumentRoot /www/company1/
<VirtualHost 1.1.1.2>
ServerName www.company2.com
DocumenbtRoot /www/company2/
</VirtualHost>
```

If you launch the first, it will only bind to IP address "1.1.1.3". The second one, since it has no "BindAddress" directive, will bind to the port on all IP addresses. So, you want to launch a server with the first set of config files, then launch another copy of the server with the second set. There would essentially be two servers running.

PidFile

This is the location of the file containing the process-ID for Apache. This file is useful for being able to automate the shutdown or restart of the web server. By default, this is "logs/httpd.pid." For example, one could shut down the server by saying:

```
cat /usr/local/etc/httpd/logs/httpd.pid | xargs kill -15
```

You might want to move this out of the "logs" directory and into something like "/var," but it's not necessary.

Timeout

This directive specifies the amount of time that the server will wait in between packets sent before considering the connection "lost." For example, "1200," the default, means that the server will wait for 20 minutes after sending a packet before it considers the connection dead if no response comes back. Busy servers may wish to turn this down, at the cost of reduced service to low-bandwidth customers.

 Table of Contents

 04 - Getting Started with Your Web Server

 06 - Managing an Internet Web Server

Chapter 07 - Creating and Managing an Intranet

No matter what a company's business, employee information-sharing methods, as well as the criteria which determine to whom such accessibility is granted, will always be top priorities. From company newsletter-type gossip, to high-level proprietary engineering data, the power of htmL and Apache can help to make the search for such material less expensive, less time-consuming, and just easier all around.

Nevertheless, it can be confusing and, initially, complicated to set up an enterprise-wide Web site. This chapter will demonstrate how to set Apache up in your company, as well as how to secure it from use by unauthorized outsiders; following that is a discussion of maintaining your new Web space and adding some useful features.

In this chapter, you learn:

- Why your company should have an internal Web server
- What kind of hardware and software will be needed
- Which browser(s) are best to standardize on
- How to relocate your existing documents and format new ones
- How to add documents to your Web space without losing organization
- How to add some useful features to you internal Web server
- How to analyze and address security concerns
- How to safely share your information with other companies

Benefits of an Intranet

Every company already has in place some kind of method for distributing

information among its employees: from bulletin boards in the cafeteria, to the sending of overly numerous memoranda (resulting in automatic placement in the so-called "circular file" more often than not), to old-fashioned weekly (or even several-times-a-day) meetings which, by definition, cannot be held at the absolutely most-convenient time for all in attendance. Some have attempted to utilize their networking software to ease this burden, but it is either too simplistic for such purposes, or not compatible across platforms.

Software companies have made great strides in the past few years; a lot of vendors are claiming to have incorporated "Workgroup Technology, " "Groupware, or "Document Sharing." Upon closer examination, however, one quickly determines that these systems are proprietary by nature, the result being that one needs an expensive server, in addition to clients for each machine and, of course, not all platforms are supported by every client. Some packages require fundamental network changes in addition to added software and hardware requirements.

Fortunately, along came the World Wide Web, originally designed to be an easy way for scientists to share ideas. It is based on a simple and open data format (htmL) and a common network protocol (TCP/IP). This means it is easy to set up and operate and, due to the open data format, there is a plethora of clientele. An *Intranet* is basically an internal Web site, though the term is also used to describe other supporting programs such as Email or UseNet news.

Using Apache(and any WWW browser), you can easily set up many effective ways to share information, communicate ideas, and exchange tips. We will cover a few ways in which developing an internal Web server can help.

Bulletin Boards

One of the more popular uses for internal WWW servers is as a bulletin board system. This, much like the traditional corkboard, allows people to add notices and information for everyone's use. Unlike the older paper system, though, the htmL bulletin board can be set up to be searchable, which, of course, puts much more data at one's nearly instantaneous disposal.

Using CGI scripts or writable documents, it is easy to create an attractive, friendly bulletin-board system which can be used by the whole company. Another advantage the electronic bulletin has is that it can be shared over the network. Users can be in the same building or in a different country; as long as they have network access, they can see your notices.

Information Center

An internal Web server also functions well as an information clearinghouse. Most organizations have many documents that need to be

available to employees, such as employee handbooks, phone listings, documents setting out company policies, etc. This is usually done by printing out the information and delivering it separately to each individual employee in the company or group. This is expensive and time-consuming, and is easily replaced by a series of htmL documents. This saves time because the documents no longer need to be individually distributed and, since the documents remain in electronic form, the company will save money on printing costs. Using htmL also allows searching and hypertext-referencing and can easily be changed with a simple text editor.

In addition to policies and handbooks, htmL can be used to distribute company information or industry news. A company newsletter can be as simple as a single htmL page, or it can have multiple pages and contain references to information stored across the Internet. It can have back issues archived and completely searchable. Industry news can also be as simple or as complex as deemed necessary.

It is also possible to have an electronic phone or e-mail list containing all the names, addresses and extensions; users would then search using the browser's search feature, or it may include a form to query a search engine.

Documents To Add to the Information Center
Employee handbooks
Policies and procedures
Phone or E-mail lists
Company Newsletter
Industry News

Common Forms

Common business forms can also be added to the Web server. This allows the owner of the document to modify it and make changes instantly available to form users. It also makes it easier for employees to find the form for which they are looking.

In some cases, the form can be handled automatically by using CGI scripts, or forwarded to the correct people, directly from the client's browser.

For more information on CGI scripts and CGI security see Chapter 13, "CGI Scripts and Server APIs."

CAUTION

Automating scripts can be a time-saver, as well as a security problem. Think carefully before automating a form request. Scripts that don't carefully check what they are doing can cause damage by executing commands improperly.

Forms To Add to the Web Server
Equipment request form
Vacation request form
Network change requests
New user account forms
Support desk trouble ticket forms
Software or hardware bug reports

Workgroup Server

As previously mentioned, the WWW was originally designed to help share information among different research groups located within various organizations. It can also be used to share information between or among workgroups.

Workgroups can use the power of the Web to list current projects, past activities, and planned proposals. This allows different groups to contact people who may have already discovered how to avoid a certain problem, or found a better way of doing something.

Team members can set up a central area for all the documentation necessary to a project, and then make allowances for easily adding individual comments. This could be very helpful when designing a new project - for example, each designer could add his or her input to the design as it goes along. This would allow discussions about the document to take place in the body of the document itself and be recorded, along with the document, for others to see.

This documentation can also include any notes, memoranda, reports and studies related to a particular area. The htmL could be searchable, making it easy for engineers to have a wealth of information right at their fingertips when encountering a new situation. This would be helpful for the novice employee, as well as the veteran engineer.

Workgroups can also use a Web server to track project status. The lead developer could create a page showing when things are expected to be done, other developers could add comments to such a file, and everyone would always be up to date with scheduling changes.

Workgroup servers can also be used to introduce new team members and help them to become more comfortable in their new positions. It can also help establish friendly working relationships between disparate groups by serving as an introduction center.

Workgroup Uses
List past, present and future projects
Store related documentation for each project
Store discussions on documents or specifications
Track project status
Create a friendly working environment
Ease the introduction of new group members

Discussion Forum

You can also use the World Wide Web in conjunction with a news server. News servers fit in nicely with Web servers and allow easy discussion between users. Some browsers allow reading and posting news articles without the need for a separate program. Other browsers may require a separate program for posting news. Browsers are covered in more detail later in this chapter.

Internal newsgroups allow open discussion on many topics. Human Resources can answer questions about insurance benefits or vacation time, engineers can discuss current problems or new products, etc. A local newsgroup dedicated to specific software can eliminate hundreds of hours of needless labor, since users can help each other out and avoid the need to call the support desk. The newly hired can get the feel of the company and its tools by reading and posting questions.

Newsgroups are set up in hierarchical form on the Internet; the same can be true for internal use. For example, one may want to set up a news hierarchy.

Sample News Hierarchy	
local.eng	Discussion of engineering topics
local.Windows	Discussion of Microsoft Windows
local.outages	Notification of outages
local.specs	Discussions about various company specifications
local.policy	Discussion about corporate policy
local.misc	General discussion

You might want to set up a simple news hierarchy at first, and then expand as groups get filled up. For example, if "local.eng" were getting a lot of messages covering hardware and software issues, one could split the group into "local.eng.hw" and "local.eng.sw". Further division into "local.eng.hw.proj1" and "local.eng.hw.proj2" might later be to everyone's advantage.

News-server software can be downloaded and used without charge from many Internet sites, and can be set up on the same machine as Apache. Larger sites may need to separate the Apache server from the news server for more efficacious performance.

Uses for an Internal News Server
Forum to discuss software packages
Forum to ask policy questions
Notification of upgrades or downtime
Hardware and software discussion
Workgroup discussions

Support line

NOTE

There are several free news servers available on the UNIX platform. Some of the more common ones are Bnews, Cnews and the more popular INN (Internet News). If you are setting up a new news server, you will probably want to use INN. You can ftp it from various sites such as ftp.uu.net in the /networking/news/nntp/inn directory.

Monitoring Tool

The World Wide Web can also be used to allow users to see how things are running. Using Apache and Server Side Includes (SSI), or CGI scripts, you can create dynamic pages. These pages can be used in a variety of ways, such as a print queue or network status monitor.

You can create a page to allow users to check the status of their print jobs from their WWW browsers. While most operating systems allow a user to check a print queue, setting up an htmL page makes it easy for everyone's use, as well as eliminating the necessity of having different commands for different machines.

In a large network environment, parts of the network might be having problems which can cause abnormal behavior. Rather than having each and every user call the support desk, you can set up a page for checking various parts of the network and assessing status.

Using the WWW for Monitoring
Print queues
Network status
Who is logged in
Which machines are in use and busy
Available file systems

htmL as a User Environment

There are cases in which one would want to create a hypertext front end

to system software. These might include word processing, database querying, data entry, and order processing.

Many companies are starting to migrate their software to be htmL-compatible; with Java and CGI, almost any application can have such an interface. This will require users to learn only one interface, regardless of the underlying system.

NOTE

SATAN is a network security-testing package that received a lot of attention in 1995. It allows users to test for known weaknesses in the software used by Internet machines. One of the things that the developers did was to create an htmL interface. Using any browser, you could check or configure the tests, or check the status of a test.

There is also work going on in developing a low-cost (under $500) machine consisting only of a network interface, a keyboard, the CPU, and graphics display. This machine would download htmL or Java applets and display them on the screen. This would allow network managers to standardize on a common interface (htmL) for all the company's computing needs. It would also allow companies to replace overpriced PC's with the newer, more affordable "Net computer".

Choosing the Network

Apache is designed to run over a TCP/IP network - whether this network is the Internet, or a private network, makes no difference to the server; however, some clients may not understand this networking protocol and, therefore, will need additional software to work properly with Apache.

TIP

It is possible to view simple htmL files without a network or a server. Most browsers allow you to open a local file and view it. This allows you to follow links and view the page, but it does not allow some of the more advanced features, such as CGI scripts or SSI programs.

About TCP/IP

TCP/IP is actually two layered networking protocols: TCP and IP. IP stands for Internet Protocol and, as the name implies, is the protocol used by the Internet. TCP, the Transmission Control Protocol, runs on top of IP and handles the sending of data between the machine and the network. IP can also run on top of PPP for dial in access (in which case it would be TCP/PPP), but is still commonly referred to as TCP/IP.

You have probably used TCP/IP before and not been aware of it. Applications such as FTP (the File Transfer Protocol) and Telnet use TCP/IP, although it is user-transparent. You might also be familiar with the Domain Name System (DNS) or Network File System (NFS); these, too, work over IP, using UDP instead of TCP.

TIP

TCP is a reliable stream protocol. This means packets are guaranteed to have been received correctly and in proper order. UDP, on the other hand, simply sends packets. The software using UDP must make sure that the data gets to its destination intact.

One of the main advantages TCP/IP has over other networking protocols is its interconnectivity. It was designed to be used with almost any type of cabling; for example, fiber optic, twisted pair, coaxial cable, or even wireless networks are possible. Additionally, it can be used by any networking topologies, including Ethernet, Token Ring and FDDI. This makes it possible to use TCP/IP in practically any networking environment.

Fig. 7.1 - TCP/IP is a layered protocol.

TCP/IP is also machine-independent. UNIX machines can talk to other computers, printers or network devices, just as long as they all understand TCP/IP. The other machine doesn't need to be using the same OS and it needn't be the same type. This has helped to make TCP/IP one of the most popular networking choices around.

TCP/IP is an open networking protocol which has allowed many vendors to develop applications and systems that incorporate TCP/IP in them. UNIX vendors have been incorporating TCP/IP for years and it is hard to find a UNIX machine without it. It is not, however, as common on the Macintosh or PC platform, and additional software may be required to set it up. Windows 95, Windows NT, and Macintosh System 7.5 all include TCP/IP networking without third-party applications.

Sidebar

TCP/IP Terminology

IP address is the machine address. It consists of four numbers, ranging from 1 to 254. These numbers are called *octets,* and the entire address is referred to as a *dotted quad.* As an example, let's use 10.32.21.199. This number actually consists of two parts: the network number and the host number. The network number is usually the first 2 or 3 octets in the IP address, and the machine number is the remainder.

Netmask determines how much of the IP is the network address and how much is the host address. For example, the netmask of 255.255.0.0 tells us that the first 2 octets are network numbers, and the last 2 octets are host numbers. Sometimes, the netmask is referred to as the "subnet mask."

Networks are conventionally split into 3 classes, A, B, and C. A class A network has a single octet for a network portion and three octets for the machine portion. Class B networks have two octets for both the machine and network portion, and Class C have three octets for the network and one for the host. The previous example uses a class B network.

Broadcast address goes along with the netmask. It tells the machine how to talk to everyone on the local network. It is usually the inverse of the netmask plus the network number. For example, if our IP address is 10.32.21.199, and our netmask is 255.255.0.0, then our broadcast address would be 10.32.255.255.

Gateways or Routers are alternative paths to a different network. They are required if going to a network other than the one in your IP address. In complex networks, there may be many gateways; in most networks, there is a default gateway. This default gateway gets all traffic not destined for the local network.

Using TCP/IP Software

Configuring TCP/IP is not any harder then configuring any other network protocol, such as Novell Netware's IPX, or Microsoft's Netbios. In fact, once a few essentials are understood, it may be that TCP/IP is easier to configure and, if one is experiencing any difficulty, there are many consultants or administrators available who are familiar with TCP/IP.

If you need TCP/IP software you may want to look at some of the commercial Internet Suites in the next section. Most of these suites contain all the software needed to start using the network, including e-mail, FTP, a Web browser and a Telnet package.

There are also shareware and freeware versions available on the Internet and various bulletin boards. Gathering all the networking software and clients for each protocol (Web, mail, FTP) can be time-consuming; sometimes it turns out to be less expensive to go with a commercial package.

Another nice thing about the commercial packages is the support. If you are having a problem with a public domain or shareware product, it is usually possible to get an answer from someone, but requires much more dedication and patience than with supported software. With commercial software, of course, it is much easier to get the answers.

There are two ways to add TCP/IP to a machine: either use TCP/IP as the only protocol, or use multiple protocols on the same machine. It is generally much easier to use just one protocol, but if you are already using networking software that may not be an option.

NOTE

If your current networking software allows it, you may be able to encapsulate your existing network in an IP packet. This makes it easier to configure your networking software. Check the documentation to see if this is possible.

Setting up TCP/IP on a Windows machine will require a WINSOCK.DLL file to be installed and configured. If this is the only networking protocol, then that is all that is needed. If running different networks from this machine (such as Microsoft networking or NetWare), some experimenting will be necessary in order to get both protocols to work side-by-side. Your networking software documentation should "walk you" through setting up multiple protocols.

If you purchase a commercial suite of products, they will include a WINSOCK.DLL. Using a browser separately will require a Winsock package such as Trumpet Winsock or else a commercial TCP/IP stack.

NOTE

The WINSOCK.DLL file is the TCP/IP stack on Windows.

Trumpet Winsock

The most widely used shareware Winsock package is Trumpet Winsock. It is available from many FTP sites but only for the Windows environment. Although lacking the TCP/IP clients, it does incorporate support for TCP/IP over a network using a packet driver, or over a modem via PPP or SLIP. Instructions for downloading a packet driver are distributed with Trumpet. If using a separate browser such as Mosaic or Netscape, this ought to come in handy.

TIP

A packet driver is a software driver for a network board which allows networking software to work "as- is" with any network board or topology. It usually requires a line added to your AUTOEXEC.BAT file.

If using other network products, such as NDIS or ODI, it is still possible to use a packet-driver package by downloading the ODIPKT or

NDIS_PKT packages. These packages are called "shims" and allow making your ODI or NDIS software look like a packet driver.

NetManage

NetManage makes a series of TCP/IP applications, ranging from Newt, a TCP/IP stack, to the Chameleon Desktop, a full-fledged Internet suite which includes NFS client and server, X Windows emulation, and almost any Internet application conceivable.

NetManage's TCP/IP stack has support for LANs, and modems using SLIP, CSLIP or PPP. The software runs under Windows 3.1, Windows 95, and Windows NT.

NetManage also has many different packages, so one is only required to pay for what one needs.

FTP Software

FTP software also has many different application suites, ranging from PC/TCP, a TCP/IP stack and some basic applications, to OnNet, a full Internet package, including NFS client and server, E-mail, UseNet, and WWW.

PC/TCP runs on Windows 95, NT, 3.1, and Windows for Workgroups. It also allows DOS applications to have access to TCP/IP. The TCP/IP stack can be used over LANS as well as modems.

MacTCP

If on a Macintosh and looking for a TCP/IP stack, you will probably use MacTCP. MacTCP is a commercial product and contains many clients such as Telnet and FTP. The later versions are very stable, though early versions were known to have problems.

Other TCP/IP packages

There are many other companies making TCP/IP software for PC and Macintosh clients. Look around and find one that has just the utilities necessary. When purchasing a TCP/IP stack for Windows, make sure you are getting a TCP/IP stack that is Winsock- compliant.

Configuring TCP/IP

Configuring TCP/IP either on a Macintosh or on a Windows machine requires a few parameters to be entered, including IP address, netmask or subnetmask, broadcast address, and gateway or router. Some installations might also ask for DNS server, which is used to allow you to use

machine names, instead of having to remember their IP address.

CAUTION

If you aren't sure of the answers in the configuration, check with your network manager. Incorrectly setting these not only causes your machine to communicate improperly, but can also prevent other machines on the network from properly working.

Choosing a Browser

Among the benefits of an Intranet over an Internet is the fact that the company can standardize on a browser. This allows the Web designer to take advantage of some advanced features, such as Java, Frames or special data formats.

Since the browser is visible to all users, it is important to choose one that has the necessary features and, at the same time, will run on all the available platforms. In the following section, we will cover the more popular browsers such as Mosaic, Netscape, and Lynx, as well as browsers from some of the TCP/IP bundles.

TIP

If you don't get one of the bundled packages and instead get just the browser, you will need to get a separate TCP/IP stack.

Current Web browsers support many different things, including form support, different image formats, Frames, Java, Imagemaps, client-side imagemaps, various htmL tags and many other protocols. Fortunately, not every company needs all these features. Some of the more helpful features are:

1. Forms. Forms are required to submit data. Most, if not all, current browsers support forms, though some may display them differently than others.

2. Image formats. Different formats include gif, JPEG, PCX and XBM. gif is supported in almost every browser, with JPEG running a close second. There are also interlaced gif and JPEG formats, which start out blurry, but get clearer as the image comes in. Interlaced images are nice, but are by no means the most important feature. XBM and PCX files are used mostly on UNIX machines and PC's, respectively.

3. htmL tags. Some companies have added their own extensions to

their browsers to make up for a lack of features in the htmL specification. Some of the new tags allow centering of text, blinking, or changing colors of fonts or backgrounds.

4. Frames are a new feature currently added to Netscape browsers. These allow the Web designer to split the browser display screen into smaller sections. These smaller sections are called frames. If used properly, frames can make navigating the Web much easier, but if used improperly, they clutter the screen and make it unusable.

5. Java is a new language designed by Sun Microsystems Inc. It allows the Web designer to add "executable content" to a Web page. Java can be used to offload some of the CPU load off of the main server, and is much more sophisticated than mere CGI scripts.

6. UseNet news and e-mail access allow a user to send and receive e-mail and news articles from within the browser. Some browsers allow reading news and posting e-mail, but not posting news and reading e-mail. Others allow both.

7. Imagemaps and client-side imagemaps allow the Web designer to have a graphical navigation tool. They can be used to display a floor plan and allow the user to click an area, such as an office, and obtain information about the group or person who uses it. Client-side imagemaps are imagemaps that do the processing on the client side instead of the server, thus reducing the load on the server.

NOTE

The browser market changes very quickly. before deciding on a browser, recheck the market for new browsers or enhancements to existing browsers.

The Mosaic Browser

Mosaic was the first graphical WWW browser and handles the common platforms (Windows, Macintosh and UNIX), so they shall be discussed first.

Mosaic was developed at the National Center for Supercomputing Applications (NCSA) at the University of Illinois in order to handle the recently created WWW. Mosaic was the first browser to include images, sound, and text in a browser environment (see fig. 7.2).

Fig. 7.2 - Mosaic captured most of the early browser market.

The latest version has a built in e-mail sender, hotlist manager, and news reader. It also has support for forms, imagemaps, and most htmL tags. It does not, however, allow posting of news or reading of e-mail. It includes support for inline viewing of gif, JPEG and XBM. Mosaic must download the entire page before it is displayed and, therefore, has no support for interlaced images. The newer Windows versions require a 32-bit version of Windows such as Win95 or NT. You can download a 32-bit DLL for earlier versions of Windows from Microsoft.

Netscape's Navigator

Netscape Navigator is considered to be the leading browser and now claims over 70 percent of the browser market. It has versions for Windows, Macintosh and UNIX (like Mosaic) but the current versions are no longer free. The last free version was 0.9 and could be used for noncommercial uses only. Navigator 2.0 has many new features, such as Java, Frames, extended htmL tags, client-side imagemaps and interlaced images. See figure 7.3.

NOTE

Netscape has added some extra tags to allow for more formatting options. These include tags to center text, change text size or color, cause sections of text to blink and also allow background images. These are called Netscape extensions.

Fig. 7.3 - Macmillan Computer Publishing's site viewed with Netscape.

Netscape has support for a complete e-mail system and can filter mail by subject, date, sender or size. It allows the user to do almost anything that can be done in a dedicated e-mail package and allows hypertext to be followed with a click of the button (see fig. 7.4).

Fig. 7.4 - Netscape's e-mail system.

In addition to the e-mail system, there is also a built-in news system. It allows both reading and posting and can be sorted by subject, date and sender. The news system also allows hypertext linking. This hypertext linking helps make Navigator one of the easiest-to-use Internet systems around.

Another nice feature of Navigator is incremental loading, which means that as the page is downloaded, it is displayed. This allows the user to start reading the text before the entire page is read in. On a fast LAN connection, this may not be as important, but on an internal network made up of slow WAN links, this will most certainly be advantageous.

Lynx Text Browser

Lynx is a fast text-only browser, developed at the University of Kansas. Lynx is very useful in environments that use dumb terminals without the ability to display graphics (see fig. 7.5). Lynx is also useful over slow links, since it does not have to download the graphics. See figure 7.5.

Fig. 7.5 - Accessing the Macmillan site with Lynx.

Lynx has support for forms but cannot display inline graphics or imagemaps. Lynx can download the graphic and view it with an external viewer; however, the terminal being used must be able to display graphics, and the viewer must be configured separately.

Lynx allows the reading of news and the sending of e-mail, but is not a full-fledged news or e-mail system like Navigator. For sites with dumb terminals, Lynx is one of the few existing alternatives.

Microsoft's Internet Explorer

Microsoft's WWW browser is an impressive browser. It supports forms, imagemaps and supports almost all the htmL tags that Netscape does. These include centering, changing font color, backgrounds and font sizing. See figure 7.6.

Explorer also supports gif and JPEG images and audio sounds. Microsoft has decided to license the Java language from Sun Microsystems and future browsers are expected to have support for it.

Fig. 7.6 - This is what Microsoft Internet Explorer looks like.

Explorer has limited table support but has problems with some tables that Netscape can display. If standardized on Explorer, though, it is easy to work around this.

Microsoft has also announced a version of Explorer for the Macintosh platform.

Chameleon's WebSurfer

The browser that ships with the Chameleon software has all the basic support, such as gif images, and most htmL tags. It does not however, contain support for Java.

Fig. 7.7 - Macmillan's site viewed using WebSurfer.

WebSurfer also incorporates a news reader and supports sending of e-mail from the browser.

Network Topology and Speed

Most LANs (Local Area Networks) can transfer at least 4 MB of information a second, which is plenty fast enough for transferring the largest text files. If the network is made up of WAN (Wide Area Networks) links, there may occur some network bottlenecks when transferring pages over them.

If using large graphic or audio files, you will experience delays on all but the fastest LAN technologies. Large files will also take more space on the server and place a strain on the client and server machines.

WAN links are generally much slower then LAN connections, usually no more than 1.5 MB and commonly only 56 kbps. Some dial-up links are only 14.4 kbps or 28.8 kbps. If there are many such connections, it is important to be careful about page size.

Even if you have a fast network, and aren't separated by WAN links, it makes sense to place the Apache server as close to the center of the network (or backbone) as possible. This will make the speed even faster and will also increase overall network performance (see fig. 8.8).

Fig. 7.8 - The server should be in the center of the network.

If many groups are using parts of the Web server, one can separate the Web space into multiple servers to increase performance (see fig. 7.9). This allows placing the Web server as close to the main users as possible.

Fig. 7.9 - You might want to split your Web server.

Disk Space

If you are only using htmL files, then very little disk space is needed. htmL files are simply ASCII files and don't take up much room.

Once graphic or audio files are incorporated, the requirement for disk space expands accordingly. An average graphic file can take several megabytes, and some file formats can take over 30 MB per file. Audio files take several megabytes of space for each minute of sound. Using these type of formats will require not only more space, but also more network bandwidth, as discussed in the previous section.

When indexing your Web space, don't forget to consider the space required for the index database. This file can be as much as half the size of the documents combined within. Log files can also take up a lot of space. Log files up to 10 MB a month are not uncommon.

Document Formats

Most companies are made up of different departments and different machine types. Marketing groups may have Macintosh machines,

Engineering might have UNIX machines, and there is almost sure to be a wide array of PCs in use.

There are also different formats used on each machine. PC users are likely to be using WordPerfect or Microsoft Word to create files. UNIX users may use text editors or Framemaker to create memoranda, while Macintosh users may use other formats.

Deciding on a common data format is not easy and, in some cases, not possible. Next, we shall take a look at the various formats and their associated benefits.

htmL

htmL was designed to overcome the cross-platform issue, but it is limited in the formatting that can be done to documents. Netscape and others have added specific tags to help, but using these tags requires using specific viewers to properly see them.

Many word-processing companies have developed (or are developing) ways to save their documents as htmL documents. Microsoft, Framemaker, Corel and others have filters or templates that you can use to create htmL files using their software.

These converters and tools are covered in Chapter 10, "htmL Editors and Tools."

There are also conversion utilities that will convert from many formats to htmL. Not all documents can be converted easily, though, and sometimes it is easier to recreate a document than to try and convert it.

Microsoft Word

Microsoft Word is a very popular word processor, having the advantage of being available on both the Mac and PC platforms. It allows graphics and various fonts and is very user friendly.

Microsoft Word files, however, are different between versions, and the early versions can't read the newer versions' files. Microsoft has made a Word viewer which is not available on UNIX machines. Still, Word remains a good choice for those owners of Windows or Macintosh machines exclusively.

Postscript

Postscript is a very powerful document language. It is also very popular, and Postscript viewers are available for the major platforms. Almost all the word-processing and drawing packages can save as a Postscript file, making Postscript seem like a good choice.

Postscript files tend to be quite large, especially if they contain many images. While they are quick on high-end machines, Postscript viewers can be slow to load on some others. In spite of this, Postscript is another very good alternative for documents that need more formatting than htmL allows.

Adobe PDF

Adobe Acrobat files allow extensive formatting and also are available on all the popular platforms, such as PC, Mac and UNIX. PDF viewers are available free of charge over the Internet.

Like Postscript, however, PDF files tend to be large and load slowly.

Images

It is also possible to save all documents as an image format such as gif or JPEG. This would allow easy viewing on the various platforms, except for dumb terminals.

Image files are, unfortunately, very large and would use much more space and network bandwidth than other formats. Also, since image files don't contain ASCII text, they cannot be indexed or searched.

A Combination of Formats

The best alternative is to use a combination of formats to achieve the required result. Any text that does not need specific formatting (such as memoranda or notes) can be done in htmL or converted to htmL. Documents that require specific formatting can be stored in Postscript or image formats to keep their look intact.

Whenever possible, keep the number of formats to a minimum. If your company only uses PC's, then stick to htmL and Word files. If everyone has a fast UNIX machine, use Postscript and htmL.

Setting Up the Web Documents

This section will cover setting up your access-control mechanism and also go over some guidelines to help you set up some of the applications that will make your Web server even more useful.

Managing Content

When setting up documents, one will need to consider how to structure them in order to make their management easier. It often makes sense to group documents according to department. Even though Apache uses hypertext links to create an organized hierarchy of documents, one should still try to maintain a close correspondence between hypertext

links and directory layout.

It is also crucial to decide who will be able to change files and create new topics. Will anyone be able to add a new topic? Who should be allowed to change pages? Do all files need to be approved? These questions must be answered in advance of starting to add files to the server.

Organizing Hierarchy

Chapter 5, "Apache Configuration," gives more information about this directive. See.

Apache doesn't have a document control feature, but you can take advantage of the DirectoryIndex directive. Using DirectoryIndex will make it easier for you to find documents when not in a Web browser; for instance, when editing (see fig. 7.10).

Fig. 7.10 - Setting up a directory hierarchy.

One of the best ways to organize a Web site is to set up directories for each major topic in its DocumentRoot. Using the DirectoryIndex directive will set up an "index page" for each topic. That page then references other files stored in that directory, or points to a subdirectory which contains another "index page". Using this file hierarchy makes it easy to organize and maintain the Web server's content.

Getting Files on the Server

If already using NFS, there is an easy way to get your documents on to your Web server. Set up the export list on the Web server to allow developers to mount DocumentRoot to their machines.

NFS may not be desirable in a high-security environment, due to its lack of authentication. PC and Macintosh users can pretend to be any user. Most UNIX users, however, don't allow pretending to be root, but care must still be taken when exporting via NFS.

See Chapter 5, Apache Configuration."

CAUTION

Exporting a file system must be done with care. If an unauthorized user can access DocumentRoot via NFS or FTP, then any server-access control is circumvented.

If you can't use NFS, then FTP is another logical choice. FTP will allow the user to cd to the DocumentRoot directory and also to put the files in

place. Using FTP is more secure then NFS, since it asks for a login and password. It does, however, send the login name and password unencrypted across the network, which may be unacceptable in some companies. Security is covered in more detail later in this chapter.

NOTE

Both of these methods require the UNIX permissions to allow write access to the directory.

Access Control Methods

The first thing you need to do is decide on who will be able to change various documents. There are three types of access control that you can use for your Web server: Open, Distributed and Centralized. We will look at each one in detail later. First, we shall discuss a little about UNIX protections.

UNIX File Permissions

UNIX file permissions control who can do what to a file or directory. This is the only way to protect documents on your Web server, so it is important to discuss them in some detail.

UNIX permissions have three different levels of access to define who can do what to a file or directory:

- The owner of the file (User)
- A group of users who have access (Group)
- Everyone else (Other)

There are also three different things a user can do to a file:

- View the contents of a file (Read)
- Change the contents of a file (Write)
- Run the file or program (eXecute)

When the permissions are applied to a directory they have slightly different meanings:

- List the contents of a directory (Read)
- Create new files in the directory (Write)

- Access files in the directory (eXecute)

The different levels of access can be defined by a number. Read is 4, Write is 2 and eXecute is 1. To determine the permissions add up the numbers. For example to set Read and Write access would be 6 (4+2).

The UNIX command to change permissions on a file or directory is chmod ugo <file>. u is the access for the User, g is for the group and o is everyone else. To set a directory up for the user and group to be able to access and add files to a directory, called "hr" and everyone else to be able to get files from the directory we would do "**chmod 771 hr**" . To set up a file, say "policy1", for the user to be able to change and the group and everyone else to be able to read would require "chmod 644 policy1". We could also use "**chmod 755 policy1**" - in the case of htmL files, the execute doesn't matter, unless using the XbitHack directive.

See Chapter 5, "Apache Configuration," for more details about this directive.

NOTE

The XBitHack directive is used to tell Apache that an htmL file includes Server Side Includes, and can also be used to send a last-modified header.

TIP

Different versions of UNIX act slightly different. If you are having problems, consult your UNIX documentation.

Now let's apply UNIX permissions to our DocumentRoot to set up different access policies.

Open Access

This type of access control is the easiest to setup and maintain. It allows anyone to change any file in your DocumentRoot. To set this up, simply change the permissions on all the files and directories in the DocumentRoot directory tree. In SunOS4.1, you can cd to DocumentRoot (DocumentRoot is defined in the srm.conf file), and "**chmod -R 777**." The chmod command we have seen before the -R option tells UNIX to perform this recursively.

CAUTION

Setting permissions so that anyone can read, write or delete any file

makes adding documents very easy. However, it can lead to problems if unauthorized users can get access to your file system. With the open access model, these users could change or remove any file on your Web server.

Using the open-access model makes adding documents easy. Unfortunately it also tends to get very confusing, since there is no central authority helping to organize where information is found.

Distributed Access

Distributed management is a scenario where several developers jointly manage content of their area, or a single developer manages a particular area. The server administrator would delegate various permissions to lead developers of a project, while maintaining control of the home page and other areas of the server. The lead developer could then further delegate responsibilities to other developers.

Using distributed access allows the server to maintain structure and also allows users to add documents through responsible developers.

To set up a directory to be administered by a single developer requires the server administrator to change ownership of the directory to him or her. This can be done by using the chown command like "**chown richc hr-docs**". The directory permissions should be set to 711 to allow the server user (defined by the User directive in httpd.conf) to get files from the directory.

If the directory is to be managed by more than one developer, it becomes necessary to create a group with the users that can control this area and set the permissions to be 771.

Any files created can be set so that only the owner can change them (chmod 644)or so that someone in the same group can change them (chmod 664). This will depend on how your policy is defined. The important thing is to make sure that the server-user has at least read permissions on the files, otherwise they will not show up in your Web tree.

Centralized Access

Centralized access requires any changes to documents to go through the server administrator, or a central authority.

Using this access policy allows structure to constantly be maintained throughout the Web. However, since only one person can make changes, it sometimes can make it hard to add documents as well as being difficult to administer.

Directory permissions for centralized control are simple. All the files and directories under DocumentRoot are owned by one person and set so that only that person has write permissions. All files should be 644 and all directories need to be 711.

Using Multiple Access Methods

Often it is desirable to have different sections under different access permissions. It is normal to have the top-level pages under centralized control, and then have each department maintain their own Web documents. It is a good idea to set up an area for anyone to create files.

To enable users to be able to control specific files, but not to be able to add new files, one would set the directory to 711 (owned by the administrator) and have the files underneath it owned by specific users with permissions set to 644. Administrators can then create the file and use the chown command to give the file away.

Some UNIX systems don't allow the transfer of ownership to other users, unless logged in as root.

Adding Useful Features

Earlier in this chapter, we discussed some features to add to an internal Web server to make it more useful. In this section, we will discuss strategies for their implementation.

Bulletin Boards

A bulletin board can be as simple as a writable htmL page. Users can edit the file and add notices, and view the page in their browser to read them. Most browsers allow searching the current page for keywords.

A better way would be to create a fill-out form and use a CGI to add a link to the main bulletin board page, and also create a separate page for the notice. The main page should have a list of notices and a subject, and also a search page. The search page should allow users to search by subject, date, author, or full text. See figure 7.11.

Fig. 7.11 - A Bulletin board page.

Handbooks and Newsletters

Creating or converting an employee handbook can be a good way to save money. It allows changes to be made easily and whenever needed,

instead of waiting until the next reprint. Using search capabilities can help save employees time by allowing the computer to search through the text.

First, create the Table of Contents page with links to the other pages. Each chapter should be in a separate directory and each section should be a separate page. Use the DirectoryIndex directive to make things easier to understand. See figure 7.12.

Fig. 7.12 - A sample table of contents.

If your company is using Frames, you can create a special frame for the table of contents so users can easily navigate through the manual.

For more information about search capabilities, see Chapter 15, "Search Engines and Annotation Systems."

Create a search page to make it easy to find references to specific topics.

Adding Business Forms

Forms are covered in detail in Chapter 12, "htmL Forms."

Adding business forms is just like adding any other type of form. Try to make the electronic version look like the paper version so people won't get confused.

See Chapter 17, "Database Access and Applications Integration."

It is also possible in some cases to create a CGI script to automate the handling of the form. This can be as simple as e-mailing someone the information on the form or as complex as adding a record to a database.

CAUTION

Take care when having your CGI make changes - it is possible to have an incorrect script remove records or make unintended changes. It might also be a security problem if users can make changes to which they should not be entitled.

Workgroup Pages

Setting up an area for a specific department can make the WWW a very powerful tool. Different departments might have different ideas on what they will use their area for, but a few common uses are: to track project status, to store documentation (knowledge base) and to introduce the members.

Tracking project status can be done by creating a page with deadlines on it. There should be a form for developers to submit changes such as missing a deadline or finishing part of the project early.

Having all the documentation, notes, meeting minutes and memoranda for a project in a central area makes communication easier.

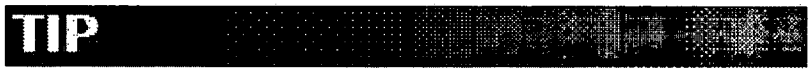
Setting a directory aside for each project and creating a search capability for it will allow employees to spend more time working on a project and less time searching for relevant information.

A Web page is also a good place to put pages about each team member and what that person's specialty is. Contact information such as phone numbers or e-mail addresses is also helpful.

Discussion Forums

There are several ways to get a discussion area created. The first and easiest way is to create a writable page and have users add comments to it. This, however, is not very interactive or easy to use.

A better way is to use UseNet news groups to allow discussion. Setting up a news server is easy and allows many different discussions to be going on at once.



Most browsers allow you to read news by using a special URL. news:local.eng.sw would generate a list of news articles in the group local.eng.sw.

Some browsers also allow posting from inside the browsers. Other only allow reading. If your browser allows sending e-mail, but not posting news, you can get a news-to-e-mail gateway and still use newsgroups for discussion.

Mailing list software often has a news gateway. One of these is listproc. Listproc can be downloaded from **ftp://cs-ftp.bu.edu/pub/listserv**.

There is also a PERL script called mail2news which can be found at **ftp://relay.cs.toronto.edu/pub/moraes**.

Monitoring Tools

It is often desirable to be able to check on the status of a print job from a Web browser. Using CGI scripts or SSI (and a little creativity), this can be easily done. See figure 7.13.

Create a network status page that contains a list of printers along with how many print jobs they have in them. You could also list who owns the print jobs. Under SunOS, use the lpc and lpq commands to get this information. Other versions of UNIX may use other commands.

Fig. 7.13 - Pages can show the status of all the print queues.

For more information about SSIs, see Chapter 13, "CGI Scripts and Server APIs."

Here is a simple script to list who is using the printers. It is to be used as a Server Side Include.

Listing 7.1 Showing the Status of Print Queues

```
#!/bin/sh
# This script is an example. It is to be used as a SSI.
#
# Cycle through the printers and get the printer name
# and number of jobs
# This next line gets all of our printer names from
# /etc/printcap. This way the script is always current.
for name in `lpc stat all | grep : | sed 's/://g'`
        do
# In our printcap we have short names and longer names
# these are from our printcap file
# NiceName is the third name.
#lp|line| The Line Printer:\
#ls| lascii| QMS 860:\
#lw|lpost| NEC Silent Writer:\
#lf|lproff| QMS 860:\
        NiceName=`grep "^$name" /etc/printcap |
      awk -F\| ' { print $3 }'
      | awk -F: ' { print $1 }'`
        echo  $NiceName '<p>'
        echo '<PRE>'
        lpq -P$name
        echo '</PRE>'
        done
```

In your network status page, it is good to include a list of machines on the network - placing a green dot if the machine is accessible, for example, or a red dot if it isn't. Using the ping command, one can check to see if the machine is alive or not.

Here is a simple SSI that can be used to test to see if a machine is running or not. It needs a graphic called reddot.gif and greendot.gif. These will be used to show if a machine is reachable on the network or not.

Listing 7.2 Testing Machine Response

```
#!/bin/sh
# Ping each machine and see if they answer
#
# ping might be in a different spot on other UNIX versions.
# This is OK for SUNOS
PING=/usr/etc/ping
# Time to wait in seconds before deciding a machine is
# really down. If TIMEOUT is
# too long your SSI will take a long time to load.
TIMEOUT=2
for name in machine1 machine2 machine3
        do
        if [ `$PING $name $TIMEOUT | awk ' { print $3 }
            ' ` = "alive" ]
        then
        echo '<img src="/images/greendot.gif"
            alt="Alive"> '$name'<BR>'
        else
        echo '<img src="/images/reddot.gif"
            alt="Down"> '$name'<BR>'
        fi
        done
```

TIP

Ping checks the network connection to a machine. It does not check to see if the machine is really working properly. Most versions of ping allow a time-out value. This is important since ping normally waits many seconds before realizing a machine is down. This would cause your page to load very slowly.

It is possible, using SSI and CGI programs, to make almost any UNIX command into a useful page. These examples are just possibilities, and not a definitive list of what is possible.

Protecting Your Data from Outside Access

Once you decide to develop an internal Web site, you need to make sure outsiders have no access it. This should be one of your top concerns. This section will cover the various ways in which one may protect data.

Security Through Obscurity

One of the easiest ways to reduce the risk of someone getting your internal data is simply to fail to inform them it is there. This is called "security through obscurity" and is the least effective method. If it is important to keep out anyone except the most casual browser, use better security.

The first way to try to hide your server is to give it an unusual name. Most companies call their Web server www.company.com, and that is the first place any determined cracker will look.

CAUTION

If trying to hide the server name, it is important not to post news, send e-mail, or run a Web browser from it. This may cause your machine name to show up in other system's logs. These logs can be set up to get automatically indexed by a search robot.

For more information about the inetd.conf file, see Chapter 5, "Apache Configuration."

The second way to make your server less likely to be found is to run it on a nonstandard port. Ports can range from 0 to 65,535, so there is a wide range to choose from. Generally, the first 1024 are considered reserved ports. Make sure the server isn't running on a port that is already in use. The Port directive (in your httpd.conf file) defines what port you are running on if you are not running from inetd. If you are running your server from inetd, this is defined in your inetd.conf file.

NOTE

The term *reserved port* means that only root is allowed to run a server on it. It does not mean that the port is in use. Reserved ports have no meaning on a PC or Macintosh, since anyone can run on any port with those platforms.

CAUTION

Hiding the server will not stop anyone determined to access your data. There is software available that can find any server, regardless of how obscure its name, by searching every possible IP address to which it is assigned. Running on a port other then 80 will slow people down, but there also exists software that will scan all 65,535 ports in a few minutes.

Using the Software To Restrict Access

Chapter 5 covered different security features, including the `limit` directive. This directive can be restricted to serving only documents within specific IP address ranges.

To use the limit directive as such server-access restriction, set the limit directive to only allow gets or posts from your IP address range. For example, if your network is 10.32.21.0, then your limit directive should say "allow from 10.32.21.*" and "deny from all".

CAUTION

Using the Apache server to protect access only stops attacks against your Web server. It does not prevent people from getting your data through other methods, such as FTP or NFS.

It is possible for intruders to trick your Web server into thinking it is part of your network by using IP spoofing. IP spoofing is a means to make a remote machine appear to be from the trusted local network.

Another software product that will help to protect a server is called *TCP wrappers*. It can be downloaded from the Internet and is available free of charge. The wrappers allow defining of who can connect to various ports and also offers some protection against IP spoofing. They also offer logging, and can be used to send different messages to different IP addresses. TCP wrappers can only be used if you are running from inetd.

Firewalling the Internet

Firewalls are the best defense from the Internet. There are different types of firewalls and each has its advantages and disadvantages. We will cover the basic design philosophies and how to use them.

CAUTION

The Internet can be a source of unauthorized access, but it is not the only way intruders can get in. Modems on people's machines can unintentionally be set up to allow crackers to get into the network.

Screening Routers

Using routers to block access to all machines and ports is one of the more common ways to protect your internal network. You can use a deny-all policy or an allow-all policy.

The deny-all policy says, in effect, not to let anything in except what services are necessary. This is the most secure method, since one can be fairly confident of what gets into your network. For example, if it is desirable to allow e-mail to get into the network, allow port 25 (smtp). See figure 7.14.

Fig. 7.14 - Screening routers can be used to allow some protocols to pass while refusing others. Http can pass, Telnet cannot.

The allow-all policy defines certain services that won't be allowed. For example, to prevent people from logging in, one would deny Telnet and rlogin services. This is not as safe as deny-all, because some services may be overlooked, leaving a backdoor open.

Using screening routers, it is possible to restrict certain types of traffic to certain specific machines; for example, allowing e-mail to every machine may not be necessary, in which case, one would limit e-mail access to only the one machine.

It may also be desirable to limit who can access the Internet from inside the company. It would be possible for employees to, knowingly or not, transfer information outside of the company. Blocking access to the Internet is discussed in a later section.

Always limit as much as possible, to reduce your risk. Only allow the minimum services through your router that are absolutely necessary and, then, only to the machines that must have them.

Application Gateways

Instead of using a router to pass or deny traffic, it is possible to have programs that decide whether or not to allow specific commands. These are commonly called application gateways, since they are specific to the application they are running (see fig. 7.15).

Fig. 7.15 - Traffic is kept separate using application gateways.

An example of an application gateway would be a proxy server. Proxy servers allow connections on each side of a network to communicate to each other, but the traffic can be analyzed and limited. For example, using a proxy Web gateway, one might allow GET access but not POST access.

Application gateways often have performance problems, since each network transaction must be checked to be sure it is not doing anything inappropriate. Some application gateways may require slightly different commands than the ones to which users have become accustomed.

Application gateways are, however, very good at auditing and logging accesses, and often are used in conjunction with a screening router.

One of the main problems with an application gateway is the fact that a single compromise makes the entire network vulnerable. Once an account

on the gateway machine is broken, it can be used to attack the internal network as a whole.

Using a Combination of Security

Most sites that are connected to the Internet use a combination of these security tools. A common firewall setup would involve one or two screening routers surrounding an application gateway. This offers some protection to the gateway host, as well as reducing risks to the internal network, if a gateway-host compromise occurs. See figure 7.16.

Fig. 7.16 - Many sites use multiple protections.

Allowing Users To Safely Access the Internet

To really take advantage of the Internet, users must be able to retrieve information from it. Many people think of the Internet as all fun and games, but there is a wealth of useful business information available, as well. This section will discuss ways to allow users to access such information from the Internet safely.

Internet Threats

There are two main threats when allowing users to access the Internet. They are: viruses, and information leaks. The risks from both can be reduced by a few simple technical solutions and by educating prospective users.

Viruses

Viruses are programs that replicate themselves to other files on the same machine and also to other machines. They may or may not be destructive, but they are most certainly always a nuisance.

Viruses can only be controlled by having users scan files on their local machines before using them. Even though all traffic may go through a firewall, it is still not possible to check all the information for viruses.

CAUTION

Even if it were possible to have your firewall scan for viruses, if people transfer files via other means (such as floppies or tapes) they are still vulnerable.

There are many different type of virus software available both free and commercially. Choose one that is most suitable and make a definite practice of running it.

NOTE

Virus software must be current to be effective. New viruses are found every day; some can't be found with older virus scanners. Always get the latest version available and keep upgrades current.

Information Leaks

There are many ways for outsiders to break into your network and it is important to reduce the risk if they do. By limiting ease of access to information from within the company, confidential data is protected in the event your site is compromised.

One of the most popular ways to limit information leakage is by using a firewall to limit who can send out information. Information can be sent many different ways, however, and can't be eliminated without disconnecting from the Internet entirely.

The easiest way to limit exposure is to limit what information can get out to the Internet. This is done by blocking direct access out of the company at the router. To allow people to get out to the Internet, install either a proxy server or SOCKS.

Proxy servers are transparent to the user and are available for many different protocols. However, each protocol must have a separate proxy server.

SOCKS, on the other hand, is a more generic tool which can be used with any protocol; however, SOCKS necessitates some changes to the client software. Such software is considered to be *SOCKSified.*

Sharing Your Private Server

There may be times when your private server needs to be accessed from other companies. You may do this via dial-up connections, dedicated network links, or over the Internet.

Using Dial-up Connections

If users need only access the Web server infrequently, it might make sense to install a dial-in modem. This connection can be running PPP software to allow full network access, or could just be used as a dumb terminal to allow simple access via lynx.

Dial-up connections can be password-protected to eliminate unauthorized use. Using Caller ID (CID) can reduce the risk even more, by enabling one to restrict telephone answering to certain authorized numbers.

Dial-up lines are the most secure way of allowing access into your network, but are usually too slow for simultaneous multiple-party use.

Using a Dedicated Link

If sharing data among sites, you may want to consider a dedicated high-speed link. These links can be ISDN, Frame Relay, or a T1 link.

Dedicated links are only as secure as the network you are connecting to. If the other network can be broken into, so can yours. Therefore, it makes sense to protect yourself from this link, just as you would do so from the Internet in general (see fig. 7.17).

Fig. 7.17 - Treat any connection outside your company as hostile.

Using Router Access Lists

If you are using the Internet to allow other users to access your company server, or a dedicated line to another company, it might be a good idea to set up an access list on your router.

Setting up an access list which restricts access to your Web server to a specific machine, or list of machines, is a good way to reduce the risk incurred when connecting. Such risk may be further reduced by allowing only the most remote sites to connect to the port on which your server is running (check the Port directive or the inetd.conf file).

This will help to protect from access via other means, such as NFS or FTP.

Password Protection

If there is no router to aid in protection, you should at least use the Apache password protection. This will require valid user names and passwords to access via the Web server.

Passwords are set up using the Auth and Limit directives in the access.conf file. This is covered in detail in Chapter 5, "Apache Configuration".

CAUTION

Using the Web server password protection will protect you from attacks through the Web server software. It will not protect you from access by other means such as NFS or FTP.

Encryption

Passwords and access lists will provide limited protection but all of your information is still sent "in the clear". This means anyone on a network through which your traffic passes can still read the contents.

The only way to keep your information private is by using encryption. Encryption is a means of converting your data to code. For example a simple way to hide your data would be to UUEncode it. UUEncode is a program that convert files from 8 bit (binary) data to 7 bit data. This makes it unreadable unless someone has the UUDecode program.

CAUTION

UUEncoding a file is not really a secure means of encryption, since all that is needed is the UUDecode program. This program is freely available from most FTP sites and is also distributed with all UNIX machines.

Using UUEncode will keep casual snoopers from seeing your information, but a dedicated cracker can save all your information and then run it through UUDecode in order to read it. A better alternative would be to use an encryption product such as PGP (Pretty Good Privacy) or RSA.

These packages require you to encrypt the data separately and then send it. It is not transparent, so if transparency is important, hardware encryption will be required.

Hardware encryption requires a separate device to be attached, either between you and your network, or between the two networks. An example would be an encrypting router, which will automatically scramble your network data so it will be unreadable. By setting up two encrypting devices at opposite sides of the network, you can have transparent encryption (see fig. 7.18).

Fig. 7.18 - Using hardware encryption makes it transparent.

Running Internal and External servers

Most companies will want to run two Web servers, one for public viewing and one for private use. There are different strategies available to keep these separate.

One Server, Two Directory Trees

For more information about the directives covered in this section, see Chapter 5, "Apache Configuration."

The simplest way to have separate internal and external areas is to set up different directory areas on a single Web server. You can then use the Limit directive in the access.conf file to allow internal IP addresses access only to the private tree.

CAUTION

You must define restrictions for all directories, not just DocumentRoot. This includes alias and script directories. This is done by using the directives Limit and AllowOverides.

To set up the directory trees do the following:

1. Create two directories. One for internal and one for external access. They must not be subdirectories of each other.

2. Use the Limit directive to only allow internal IP addresses to get to the internal directory.

3. Disable per-directory overrides to make sure your Limit directive stays enforced. You can do this by using the AllowOverides directive. Set this to none.

Using the same server is the cheapest solution and can be used if there are no alternatives or your internal data is not that important. If the server is compromised, not only is your external data in danger, but your internal data is compromised as well.

One Machine, Two Servers

This strategy also only uses one machine and allows security for the cost-conscious. It offers slightly more protection than the previous example but is not the most secure option available.

Using this technique, you create two separate http configurations, including configuration files and DocumentRoots. This allows you to run separate server processes for internal and external accesses. You will probably want to run your external server on port 80 since that is where most people will look for it. Your internal serve can then run on any unused port.

Running separate servers allow you to configure your internal server to be more or less restrictive then your external server.

The following is how one sets up multiple servers on one machine:

For detailed instructions on setting up Apache see Chapter 4,

"Getting Started with Your Web Server."

1. Create two separate server directories, including configuration and document directories.

2. Configure your external server as you normally would, but configure your internal one with the same restrictions as the previous procedure. (Use the Limit and AllowOverides directives.)

3. Configure your Internal server to use the non-standard port. This is done in either the httpd.conf file, using the Port directive, or in the inetd.conf file.

4. When you start your httpd server, you may need to use the -d or -f flags to point to the right configuration files.

Two Machines, One Network

A better alternative to using one machine is to use two machines: one that serves the external pages, and one for internal access.

Using two machines protects your data - so long as unauthorized access remains restricted to only the one machine, then your internal and external Web pages will never both be placed in jeopardy.

CAUTION

If any machine on your network has been compromised, it is possible that all of them have been. Crackers can install sniffer programs to watch the network for passwords and store them in a file or e-mail them to the cracker. The only way to get around this is to always encrypt your traffic.

UNIX machines also can be set up to trust one another, either by creating a "/etc/hosts.equiv" file or by putting a ".rhosts" file in your home directory. Trusting a machine that has been compromised is a sure way to get broken into. Never trust your external Web server.

If one of your servers is compromised, that machine can be used to break into other machines on your network, either by installing a sniffer or taking advantage of host trust.

Two Machines, Two Networks

Having your two machines on the same network, as in the previous scenario, can be a problem if one of them is compromised.

An even better alternative is to separate the two machines by a firewall. This firewall can be as simple as a screening router or a series of routers

and application gateways. See figure 7.19.

Fig. 7.19 - Set up your external Web server outside your firewall.

Table of Contents

06 - Managing an Internet Web Server

08 - Basic htmL: Understanding Hypertext

Chapter 5 - Server Configuration

The previous chapter presented an overview of NT Servers and got us started with the WebQuest and Microsoft IIS servers. Fortunately for us, these servers use GUI-based tools for configuration and maintenance. While most servers are migrating in this direction, many are still based on configuration files. These files control the behavior, the operational parameters, of the server. The new generation of 32-bit Windows servers tend to store this configuration information in the registry, not text files.

That said, this chapter uses the old-style configuration files as a reference point. The new generation of Windows-based servers are seeking to differentiate themselves from each other and are using different terminology to express the same points. Use this chapter as a point of reference to refer to for the configuration commands of your server software.

This chapter covers in detail all standard server configuration commands and how to use them, including those related to access control, directory indexing, and MIME types. This will provide a solid base of information that you will use to understand the different configuration steps, although they may be referred to differently by each server software package. This chapter takes you step-by-step through each file and the myriad of configuration possibilities. See "Overview of Server Software," (Ch.5)

Our demonstration files come from the standard NCSA server originally created for the UNIX environment. The NCSA Server uses four primary configuration files that you may view as a set of toolboxes. With most servers you can use as many or as few features as you like. If you want to get a plain-vanilla Web server up and running quickly, the configuration files usually require very little editing. However, the power is always there to get your tools out and create the additional features that you need.

In this chapter, you learn the essentials of how to:

- Set up the server's physical network parameters
- Serve documents from multiple directories
- Use and configure server-generated directory indexes
- Configure MIME types for launching viewers
- Implement password security and access control
- Set up Windows NT and Windows 95 for Internet Networking

Overview of Configuration

This chapter presents information on using the NCSA server configuration files. Table 5.1 lists the four primary server configuration files.

Table 5.1 Server Configuration Files

File	Description
httpd.conf	Primary server configuration file
srm.conf	Server resource map
access.conf	Security configuration
mime.types	Information on document types

The WebQuest Servers use a GUI administration program and store all associated configurations in the Windows registry

General Principles

Making any server configuration change requires editing one or more of the server configuration files. All configuration files are simply plain ASCII text that you con modify in any text editor (Notepad, Wordpad, and so on). All of the files follow these rules:

- All files are case-insensitive ("ALL" is the same as "all").
- Comment lines begin with the number sign (#). Comments must be on a line by themselves.
- Except in the access configuration file, the order of statements is not important

File and path names in the configuration files must be given in UNIX format using the forward slash (/).

When you start the server, it reads the configuration files and loads them into memory for as long as the server is running. In order for changes made to the configuration files to take effect, you must restart the server (just the software, not the whole machine). This means that you can edit the files while the server is running without having any effect on the server. Only when the server is restarted do changes take effect.

Terminology

Many of the server configuration commands specify a path to a directory or file. This path can be one of two types, a physical or virtual path. In some cases, either can be specified.

A *virtual path* is the document path specified in a URL. In the URL **http://www.xyz.com/sales/intro.txt**, the virtual path is sales/intro.txt. This may or may not correspond to an actual path on the server. One of the server functions is to map a virtual path to a physical path on the server. The *physical path* is the actual location on disk, such as C:\WINDOWS\MM.BMP.

A physical path can either be absolute or relative. In Windows, *absolute paths* begin with a drive letter (like C:)or backslach (\). *Relative paths* begin with the name of a file or directory, like wedding/gowns/white.txt. Two special symbols can also be used in relative paths. A single dot (.) stands

for the current directory, and two dots (..) means the parent directory. The previous example could also be written ./weddings/gowns/white.txt. The path ../birthdays points to the birthdays directory underneath the parent directory.

(d)The Server Configuration File

The *server configuration file* is the main configuration file. By default, it resides in a file called HTTPD.CNF. This is the starting place for configuration. It contains information about the server itself, the locations of log files, and the locations of other configuration files. This section shows you how to configure the server's primary configuration file.

> If your server supports configuration files, it's a good idea to make backup copies of all configuration files before you begin editing.

Server Information

Several types of server information are specified in the server configuration file. These include parameters directly affecting the server's operation such as the server's port and time-out. Elements in the server configuration file used in error reporting include the server administrator name.

To define basic server information, you use the following directives:

- Port
- TimeOut
- User
- ServerAdmin
- ServerRoot
- ServerName

Setting the Server Port

In order for you Web server to run properly, it must know what port to listen to for information. A port is somewhat like a CB channel, but it is implemented in software rather than physical channels. All Internet traffic to the server is carried on a single network cable; there are not really multiple channels. Every information packet contains a destination port number that is read by the receiving machine so that the data can be sent to the right program. Different types of Internet traffic are carried on different ports. Table 5.2 lists commonly used Internet ports. To set the server's port, use the Port directive:

Port

- Usage: Port *number*
- Example: Port 80
- Default: Port 80

The Port directive sets the port number that the server listens to for incoming requests. Most Web (HTTP) traffic is on port 80, but it is possible to set the port to any number from 0 to 65535. Ports under 1024 are reserved for the most common types of Internet traffic, so it is recommended that you use a number above 1024 if you need an alternate port. As a rule, experimental Web servers are often run on port 8080. See "Managing an Internal Web Server," (Ch. 8)

Some network routers are configured by default to allow all incoming and outgoing Internet traffic on ports greater that 8000. It is very easy, therefore, for a PC or workstation on an internal network to bypass Internet security mechanisms by running a Web server on a high-numbered port.

Table 5.2
Commonly Used Internet Ports

Service	Port
FTP	21
Telnet	23
NNTP	119
SMTP	25
HTTP	80
Gopher	70
IRC	6667
Talk	517
Finger	79

Setting the Server's Time-out

All Internet data is sent in short bursts called *packets*. By breaking up each transmission into small packets, it is necessary to resend only a small amount of information (one packet) when there is a problem with the data transmission. Normally, a client receives a steady stream of packets from an Internet server. However, if the stream is interrupted for a significant time interval, the client or server can *time out* and abort the connection. To set the server's time-out interval, use the TimeOut directive:

TimeOut

- Usage: TimeOut *seconds*
- Example: TimeOut 30
- Default: 30 seconds

The TimeOut directive is used primarily to facilitate more reliable connections between sending or receiving successive data packets. If, due to a busy or slow link, successive data packets are not sent or received within the allotted time, the server times out and reports an error. It may be necessary to set the timeout to 60 seconds to accommodate slow PPP or SLIP dial-up links on either the server or client end.

Locations of Other Configuration Files

Because the server configuration file is the main configuration file, it defines the location of all the other configuration files, including the global access configuration file, server resource map, and MIME types

configuration file.

You can specify the names of these files as absolute system paths or relative to the server root directory by using the ServerRoot directive.

ServerRoot

- Usage: ServerRoot *directory*
- Example: ServerRoot /test/httpd
- Default: C:\HTTPD

The ServerRoot directive specifies the path to the top-level directory containing the configuration, support, and document directories and files.

Locations of Log Files

See "Usage Statistics and Reports," (Ch. 18)

Most servers can log every document access and every error. You can use several different utilities to collate and graph these statistics after they are collected.

Two directives specify the location of log files: AccessLog and ErrorLog

ErrorLog

- Usage: ErrorLog file
- Example: ErrorLog /errors/web.err
- Default: logs/error.log

The ErrorLog directive specifies the location of the file that records server errors, including:

- Documents that could not be found
- Timeouts due to slow connection links
- Connections that have been interrupted
- Script errors
- Invalid configuration files

Reverse DNSLookup

One final directive in the server configuration file doesn't fall neatly into any other category. You can use the ReverseDNSLookup directive to determine the computer names of those accessing server documents.

- Usage: DNSLookup {on|off}
- Example: DNSLookup off
- Default: off

The ReverseDNSLookup directive determines whether the server will attempt to find the user name of the remote client for every document requested. The remote user name is only available if that TCP/IP address is referenced by a DNS server. Unless you really need to know computer names, leave this

option off because it requires considerable overhead.

Most Webmasters don't really need to know the computer name of the client until it comes time to look at statistics. Today's statistic packages will look up the client names for you as they compile the statistics.

Remote identity checking is intended to be for information purposes only. It is not a secure mechanism for identifying a client. The information returned can be misleading or wrong.

MIME Types

Multimedia Internet Mail Exchange (MIME) types provide a way for Web browsers to automatically launch a viewing program associated with the file received. For example, the server MIME types file maps an image file in the JPEG format that has the extension JPG to type "image/jpeg." This type is then sent to the browser, which you can configure to launch a JPEG viewer when it receives a file of type "image/jpeg." You can use the MIME types to launch word processors, spreadsheets, or any program that resides on the client's machine, as well as to launch multimedia viewers.

The MIME Types Configuration File

A list of registered MIME types in contained in the MIME types configuration file that comes with the server. By default, this file is often named MIME.TYP.

Each line in the MIME types file contains one MIME type/subtype and a list of file extensions to map to that type. Image, audio and video types launch their respective multimedia viewers. Text types are for documents written in plain text, but can also be interpreted using formatted tags (like HTML). Application types are defined for most everything else. Subtypes containing x- are experimental types. Listing 5.1 is the standard MIME types file that comes with NCSA httpd.

Listing 5.1 Standard MIME Types File

```
application/activemessage
application/andrew-inset
application/applefile
application/atomicmail
application/dca-rft
application/dec-dx
application/mac-binhex40
application/macwriteii
application/msword                      doc
application/news-message-id
application/news-transmission
application/octet-stream                bin
application/oda                         oda
application/pdf                         pdf
application/postscript                  ai                              eps                             ps
application/remote-printing
application/rtf                         rtf
application/slate
application/mif                         mif
application/wita
application/wordperfect5.1
application/x-csh                       csh
```

application/x-dvi	dvi		
application/x-hdf	hdf		
application/x-latex	latex	ltx	
application/x-netcdf	nc	cdf	
application/x-sh	sh		
application/zip	zip		
application/x-lzh	lzh		
application/x-gzip	gz		
audio/basic	au	snd	
audio/x-aiff	aif	aiff	aifc
audio/wav	wav		
image/gif	gif		
image/ief	ief		
image/jpeg	jpeg	jpg	jpe
image/tiff	tiff	tif	
image/x-cmu-raster	ras		
image/x-portable-anymap	pnm		
image/x-portable-bitmap	pbm		
image/x-portable-graymap	pgm		
image/x-portable-pixmap	ppm		
image/x-rgb	rgb		
image/x-xbitmap	xbm		
image/x-xpixmap	xpm		
image/x-xwindowdump	xwd		
message/external-body			
message/news			
message/partial			
message/rfc822			
multipart/alternative			
multipart/appledouble			
multipart/digest			
multipart/mixed			
multipart/parallel			
text/html	html	htm	
text/plain	txt	log	
text/richtext	rtx		
text/tab-separated-values	tsv		
text/x-setext	etx		
video/mpeg	mpeg	mpg	mpe
video/quicktime	qt	mov	
video/msvideo	avi		
video/x-sgi-movie	movie		

Creating New MIME Types

In some cases, you may wish to create a new MIME type in order to automatically launch an application not defined in the standard configuration file. This is frequently useful on an internal Web server, where the appropriate viewer application for the new type is available on the network. However, it is less useful on a public Web server because browsers have to be specially configured for you new type. Typically, this means you have to tell users what the new type is and provide the view application. This is time consuming and inconvenient; therefore, use new MIME types sparingly on public Web servers.

You can use of several methods to create new MIME types for document types not defines in the default MIME types configuration file. You can edit the MIME types file directly but this is not recommended. A better method is to use the AddType directive in the server resource map (covered in the next section) to create a new MIME type for all documents on the server.

In addition you can add MIME types for files only in certain directories by using the AddType directive in *local directory access configuration files*. The "Local Directory ACF's" section later in this chapter covers this type of file.

Do not modify the MIME types file provided with the server, because you might

accidentally modify standard MIME types. If you do edit the MIME types file directly, be sure to make all changes with comments so that you can find them later.

The Server Resource Map

The server resource map file controls server content-related features, including the location of document directories, MIME types for launching viewers, and automatic directory indexing features. By default, the server resource map is located in C:\HTTPD\CONF\SRM.CNF

Establishing Document Directories

By default, all documents are served from a single directory tree called the *document root,* which is defined by the `DocumentRoot` directive. However, it is possible to serve documents from many other directory trees defining aliases to these directories in the server resource map.

You can create simple directory aliases using the Alias directive, which maps a virtual path to a physical path on the server. You also can use the Alias directive to serve documents from location on other servers; Redirect redirects requests for documents in a given directory to another location. In addition, you can serve documents from users' personal directories on a network with the UserDir directive.

The following directives apply to document directories and scripts:

- DocumentRoot
- Alias
- ScriptAlias
- WinscriptAlias
- Redirect
- UserDir

By default, all documents are served from a single directory tree specified by the DocumentRoot directive. As far as the Web server is concerned, the document root directory is the highest level directory possible. Users of the server cannot specify a higher-level directory using the .. relative path that normally points to parent directories. This is vital for security, because all Web server security mechanisms are built upon the notion that documents are located only in those directory trees explicitly defined in the server resource map.

DocumentRoot

- Usage: DocumentRoot directory
- Example: DocumentRoot c:\httpd\documents
- Default: none

The document directory is the directory in which the server looks when it receives a URL without any path information, like http://www.xyz.com/home_page.html. Only one DocumentRoot may be specified in the server resource map. You can define additional directory trees with the Alias directive.

Creating Aliases

You can serve documents from directory trees outside the document root directory by defining aliases to

them with the Alias directive. An *alias* maps a virtual path in a URL to a physical path on the server.

Alias

- Usage: Alias virtual_path physical_path
- Example: Alias /sales c:\documents\sales\data
- Default: none

The Alias directive provides a way to serve documents from directories other than the document root directory. By default, the server looks for documents in the root directory only. Using aliases allows other virtual directories to be created. In this example, if a request was received for http://www.xyz.com/sales/march.gif, the server would look for c:\documents\sales\data\march.gif. You can define as many aliases as you like in the server resource map.

> In addition to serving documents from directories other than DocumentRoot, aliases can be used to shorted long path name. For example, "c:\xyzcorp\a1division\finance\data\sales" can be shortened to just "/a1sales" by defining an alias.

Aliases can point only to directory trees accessible on the server. This includes all drives physically located on the server and those accessible via a network. Aliases cannot be used to point to other servers. This is the function of the Redirect directive.

Redirect

- Usage: Redirect virtual_path new_URL
- Example: Redirect /data/finance http://xyz.finance.com/data
- Default: none

The Redirect directive is similar to the Alias directive, but causes URLs beginning with the special path to be redirected to a new location, which can be any other site on the Internet. The Alias directive defines an alias for a physical path, whereas redirect defines an alias to a virtual path, which can be any valid URL.

> If a document has moved, you can use Redirect to point to the new location so that browsers and HTML files at other sites on the Internet do not all have to be updated to point to the new location. However, if the referring server is unavailable the browser will be unable to locate the document. Use referral only as a temporary measure.

Establishing Script Directories

Closely related to document aliases are *script* aliases, which define directories containing executables scripts instead of documents. When the server reads a document from a script directory, it attempts to run the file rather than to send it on to the client. Script directories are defined with several variation of the ScriptAlias directive, depending on which environment the scripts are written for.

ScriptAlias

- Usage: ScriptAlias virtual_path physical_path

- Example: ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-bin

See "CGI Scripting,"(Ch.15)

The ScriptAlias directive is used to define a directory as containing executable CGI (Common Gateway Interface) scripts rather than files to be sent to the client. It also creates a virtual path to the specified directory just like the Alias directive. Files in script directories are not sent to the client. Instead, they are executed and the resulting output is sent to the client. You can define multiple script directories.

WinScriptAlias

- Usage: WinScriptAlias virtual_path physical_path
- Example: WinScriptAlias /WIN-CGI/ C:/WINDOWS/CGI/
- Default: none

The `WinScriptAlias directive is identical to the ScriptAlias directive except that it identifies directories containing Windows scripts rather than DOS scripts. You can define multiple WinScriptAliases.`

MIME Types and Encoding

As discussed in a previous section of this chapter, MIME types provide a way to launch viewers and applications from within a Web browser. The server implements this by sending MIME type information with each document.

Defining MIME types

Normally, the server uses file extensions to map documents being served to their appropriate MIME types. If a file has no extension or an extension that can't be found in the MIME types file, the server sends the default MIME type, txt/html. In certain instances, you may wish to change the default MIME type. You might want to do this, for example, if you have many formatted plain text documents without extensions. To avoid renaming all the documents to end in TXT, you can simply change the default MIME type to text/plain so that the documents are viewed correctly in Web browsers. Change the default MIME type with the DefaultType directive.

DefaultType

- Usage: DefaultType *type/subtype*
- Example: DefaultType text/plain
- Default: text/html

The DefaultType directive in the server resource map specifies the default MIME type for all documents on the server.

> If your server contains many nonHTML files that have no extension, it is important to use the DefaultType directive to specify the correct document type. Otherwise, the server assumes they are HTML files (type "text/html").

You can create MIME types that are not currently registered in the MIME types configuration file with

the AddType directive. In general, you should not add new MIME types unless absolutely necessary because of the work involved in configuring clients for new MIME types. This is especially true on public Web servers. Like the DefaultType directive, the AddType directive can be used to add a MIME type just for one directory by placing it in a local directory access configuration file. For more information about these files, see the later section, "Local Directory ACFs."

AddType

- Usage: AddType *type/subtype* {extension|file|phycial_path}
- Example: AddType application/msword doc
- Default: none

Using the AddType directive is the preferred method of adding MIME types to the server (as opposed to editing the MIME types configuration file). All documents having the specified are mapped to the specified type. The extension argument can be a file name or full path name to a single file as well as a file extension. For example, the statement AddType image/jpeg sat001.dat maps all files named sat001.dat to type "image/jpeg." The statement `AddType image/jpeg /pictures/earth_view/sat001.dat maps only the file sat001.dat to type "image/jpeg." Any number of these directives may appear in the server resource map.`

`A capability similar to MIME types is *encoding*, in which the server can mark compressed documents as being encoded in a specified format. Browsers that support automatic decoding can then use the encoding type information to automatically decode the file when it is received. You indicate encoding types with the AddEncoding directive.`

`AddEncoding`

- Usage: AddEncoding encoding extension
- Example: AddEncoding x-gzip gz
- Default: none

You can use the AddEncoding directive to conserve bandwidth on a busy network; however, few clients currently support automatic decoding, so it is not commonly used.

> You can achieve automatic decoding using MIME types even on clients that do not support the HTTP encoding extensions. The types "application/zip" and "application/x-gzip" (x means experimental) are already defined in the standard MIME types file. To automatically uncompress ZIP files, users simply configure their browsers to run PKUNZIP after receiving files of type "application/zip."

Directory Indexing

Directory Indexing is a powerful Web server feature that enables the server to function somewhat like an FTP server. When the server receives a document request containing only a directory name, it can automatically generate a hypertext list of all files in the specified directory, complete with dates, sizes, descriptions, and icons. Figure 5.1 shows a directory index produced by the WebQuest server. You can also include information about the directory listing above.

Fig. 5.1 - Directory Index returned by the WebQuest Server.

Types of Directory Indexes

A simple directory index contains only the names of files in the requested directory. You can create more detailed indexes by turning on *fancy indexing*, in which file size and date information is displayed along with file names. In Windows, you turn on fancy indexing with the FancyIndexing directive.

FancyIndexing

- Usage: FancyIndexing {on|off}
- Example: FancyIndexing on
- Default: on

The FancyIndexing directive controls whether automatically generated directory indexes contain icons, file size and date information, and descriptions as well as file names. If FancyIndexing is off, only file names are listed.

> In automatically generated directory indexes, files are listed alphabetically. Unfortunately, there is no way to list files by size or date.

It is also possible (and in many instances preferable) to control individual elements of a fancy index using the directives in the next three sections.

In addition to displaying server-generated directory indexes, you can configure most servers to display a prewritten index. A *prewritten index* can be any Web document having the name specified in the DirectoryIndex directive. If a prewritten index file is present in a directory, it is displayed instead of a server-generated index.

DirectoryIndex

- Usage: DirectoryIndex file
- Example: DirectoryIndex index
- Default: index.html

The DirectoryIndex directive specifies a file name to look for when a client sends a directory URL to the server. This is usually an HTML file, but does not have to be. DirectoryIndex provides a convenient way to establish a standard for locating top-level descriptions of Web documents. For example, suppose a company has assigned directories to various departments on an internal Web server. If each department creates a home page with the name specified by DirectoryIndex, Information from a given department is easy to find. Only on DirectoryIndex directive is permitted in the server resource map.

Including Directory Information

It is often useful to include general information about a directory along with the list of files generated by the server. Most servers support this by including the contents of a README file in a directory index, if one is present. The name of the README file is defines in the ReadmeName directive.

ReadmeName

- Usage: ReadmeName *file*

- Example: ReadmeName readme
- Default: none

If a file having the name defined by the ReadmeName directive is present in a directory, it is included as a footer to the directory listing. The file can either be plain text or HTML, depending on the extension. In this example, if the server found a file named "readme," it would display the file as plain text at the bottom of the index. If it found a file named "readme.html," it would display it as HTML. Only one ReadmeName directive is allowed.

It is also possible to include information before the directory listing. This is possible using the HeaderName directive.

HeaderName

- Usage: HeaderName *file*
- Example: HeaderName header
- Default: none

The HeaderName directive works just like ReadmeName, except that the included file is displayed before, rather than after, the directory index. Only one HeaderName directive is allowed.

Adding Icons and File Descriptions

Icons in front of the file names make indexes more attractive and allow file types to be identified quickly. To add icons to a directory index, you can specify a default icon and any number of additional icons. You can define the default icon with the DefaultIcon directive.

DefaultIcon

- Usage: DefaultIcon *virtual_path*
- Example: DefaultIcon /icons/default.gif
- Default: none

Only one default icon is permitted in the server resource map.

> The default icon should be the same size as all other icons used in the directory to achieve a consistent appearance for all files.

You can define icons in three ways: for all files having a certain extension or extensions, for all files having a certain name, or for exactly on file matching a specified path. Add icons with the AddIcon directive.

AddIcon

- Usage: AddIcon virtual_path {.extension|file|physical_path}...
- Example: AddIcon /icons/image.gif .gif .jpg .bmp .pcx
- Default: none

The file type specified in the AddIcon directive can either be an extension, a file name, or the full path to

a file. You can add icons to directories using the special keyword ^^DIRECTORY^^. In addition, you should specify a blank icon for the keyword ^^BLANKICON^^. The blank icon appears as a placeholder at the beginning of the line containing column labels and should be the same size as the other icons to preserve correct column alignment. You can define multiple icons in the server resource map.

> Unlike the other directives that use file extensions (AddType and AddEncoding, for example), AddIcon requires a leading dot (.) before the extension.

Often you can specify an additional element in the AddIcon directive. This element specifies text to appear in place of an icon on nongraphical browsers. In place of the virtual path to the icon file, you can use an expression in the form (ALT,*virtual path*), where ALT is any three-letter text abbreviation for the icon.

Another consideration is that file icons can be considered informational only or can actually be part of the hypertext links to the files. To make icons part of the hypertext links, use the IconsAreLinks directive.

IconsAreLinks

- Usage: IconsAreLinks *{on|off}*
- Example: IconsAreLinks on
- Default: off

If IconsAreLinks is on, the user can click either on the file name or its icon.

Besides icons, files listed in a directory index can contain descriptions. You can add descriptions for files of any name or type with the AddDescription directive.

AddDescription

- Usage: AddDescription "description" {extension|files|physical_path}
- Example: AddDescription "Microsoft Word document" doc
- Default: none (UNIX) or HTML title (Windows)

By default, most servers use a document's title (as specified by the HTML <TITLE>...</TITLE> construct) as its description. Some servers only do this if ScanHTMLTitles is included in IndexOptions. You can specify multiple descriptions in the server resource map.

Excluding Files from Indexes

By default, a directory index displays all files in the directory. However, in many cases it is not desirable to provide a link to the parent directory (..) or to any configuration files in the directory. You can exclude these from the directory index with the IndexIgnore directive.

IndexIgnore

- Usage: IndexIgnore {extension|file}
- Example: IndexIgnore CNF #HACCESS #README #README.HTM
- Default: . (the symbol for the current directory)

You can specify file to be ignored by extension or file name. In this example, all configuration (CNF), access configuration (#HTACCESS), and #README files are to be ignored, along with the parent directory (..).

It is useful to ignore access configuration files to prevent unauthorized users from learning about your access control mechanisms. Ignoring files specified by the ReadmeName directive prevents redundancy, because these files are displayed as the index footer.

Be sure to exclude local access configuration files from automatic directory indexes to prevent possible security problems. Do this with the IndexIgnore directive.

Access Control

Access configuration files (ACFs) allow or deny permission to use various server features based on Internet address or password security. For example, a whole server can be locked out to all Internet addresses, allowing only internal use. A subscription Internet service can allow access from only paying customers by allowing requests only from their IP addresses. A sensitive document or script can be password-protected so that only authorized users have access.

In addition to providing security, you can use access configuration files to modify features in the server resource map on a per directory basis. This is possible because there are actually two types of access configuration files: the global access configuration file and per directory (local) access configuration files. The next section covers the global access configuration file and its capabilities. The section "Local Directory ACFs" at the end of the chapter explains local directory ACFs.

The Global Access Configuration File

The global access configuration file is the master file. It determines the permissions and features allowed in each directory. The global file also determines the permissions and features that local directory ACFs can override. By default, the global ACF is located in C:\HTTPD\CONF\ACCESS.CNF.

The global file is broken up into sections for each directory in question. By default, all permissions are wide open. If this is not your intended configuration, you must describe the permissions for each controlled directory tree in a separate section of the global ACF.

Each section of the global ACF can contain directives that enable or disable various security features. These security features include the ability to disable potentially dangerous server features, restrict requests based on IP address, require user authorization, and disallow per directory overrides. In addition, the global ACF can contain directives that allow certain document and index capabilities defined in the server resource map to be implemented on a per directory basis.

Sectioning

The global ACF uses a *sectioning directive* named "Directory" to break up the file into one section for

each controlled directory.

Directory

- Usage: <Directory *physical_path*>...options...</Directory>
- Example: <Directory c:\httpd\htdocs>
- Options None </Directory>
- Default: none

A pair of Directory tags designates a section of the ACF. All directives contained between the Directory tags apply to the specified directory an all subdirectories. Every <Directory> tag must be accompanied by a corresponding </Directory> tag, and nested sections are not permitted.

> Every controlled directory hierarchy must be called out specifically in the global ACF. A common oversight is to protect the document root directory, but to overlook directories accessible via aliases and script aliases. These are not protected under the auspices of the document root directory because the global ACF is based on the real paths, not virtual paths.

Disabling Potentially Dangerous Server Features

Many Web Server "Features" open the door to potentially dangerous operations that can be controlled by the global access file. These include using server-side includes, implementing automatic directory indexing, executing CGI scripts, and following symbolic links.

Automatic directory indexing is potentially dangerous because people can use this feature to search directories for files you really don't want them to see. If this feature is disabled, they have to guess the names of any files not specifically mentioned in accessible HTML files. See "CGI Scripts, Server-Side Includes, Server APIs"(Ch.15)

Server-side includes allow the server to automatically insert certain elements such as date and time in HTML documents each time the document is requested. One server-side include option, exec, is particularly dangerous because it can run any program accessible from the server.

Executing CGI scripts is as dangerous as using server-side includes because, like the exec function, scripts can execute any program accessible to the server.

You can control each of these potentially dangerous operations with the Options directive, which you can place in each section of the global ACF.

Options

- Usage: Options {None|Indexes|All}
- Example: Options Indexes FollowSymLinks IncludesNoExec
- Default: All

Under some basic servers, only the Indexes option is available. Table 5.3 lists many of the elements your server may support. If Indexes is turned off, automatic directory indexes are disabled. However, prewritten index files specified by DirectoryIndex are always allowed.

Table 5.3 Options Available Under NCSA

Name	Description
None	Disallow all options
All	Allow all options
Indexes	Allow server-generated directory indexes
Includes	Allow all server-side include functions
IncludesNoExec	Allow server-side includes except exec
ExecCGI	Allow CGI scripts to execute
FollowSymLinks	Follow all symbolic links
SymLinksIfOwnerMatch	Follow links only if the destination file has the same owner as the link itself

Restricting Access by IP Address

In many cases, it is necessary to restrict access to a directory or perhaps an entire server to only certain IP addresses. You can do this with the Limit directive, which uses a paired tag format similar to Directory.

Limit

- Usage: <Limit {GET|PUT|POST} ...>...options...</Limit>
- Default: Open to all addresses, no passwords required

The Limit directive occurs inside a pair of Directory tags and limits access to files in the specified directory tree. The first Limit tag specifies what kind of access is restricted. GET access is the most commonly used and includes all document requests from the directory. PUT access is not yet implemented in the NCSA servers. POST access means posting form data to the server for script processing. Limit supports four options inside the pair of Limit tags: Order, Deny, Allow, and Require. Order, Deny, and Allow are discussed here, and Require is discussed under user authorization.

Deny

- Usage: Deny from {*host(s)*|all}
- Default: No hosts are denied

The Deny directive refuses the type of access specified in the Limit directive to the specified host(s). A *host* can be a full or partial host name or IP address or the keyword all.

Examples:

- Deny from 127.0.0.1
- Deny from 127.0
- Deny from racecar.ncsa,uiuc.edu
- Deny from .uiuc.edu
- Deny from all

Allow

- Usage: Allow from {*host(s)*|all}
- Default: All hosts are allowed

The Allow directive allows the type of access specified in the Limit directive to the specified host(s). A host can be a full or partial host name of IP address or the keyword all.

Examples:

- Allow from 142.167.100.115
- Allow from 142.
- Allow from 142.167
- Allow from monkey.zoo.stlouis.com
- Allow from all
- Allow from.stlouis.com

Order

- Usage: Order {deny,allow|allow,deny}
- Default: deny,allow

The Order directive specifies in what order Deny and Allow directives are evaluated. This is very important when a host occurs in both Allow and Deny directives because of wild cards. By default, the order is deny, and then allow.

> The order in which Deny and Allow directives appear in a Limit section has no bearing on the order of evaluation. This is determined strictly by the Order directive.

Examples:

```
<Limit GET>
Order deny, allow
Allow from 167.142.
Deny from all
</Limit>
```

In this example, only host addresses beginning with 167.142 are allowed. All others are denied.

```
<Limit GET>
Order allow, deny
Allow from 167.142.
Deny from all
</Limit>
```

In this example, GET access is denied to all hosts, even though 167.142. is specified in an Allow directive. Why? Because the Allow directive is evaluated first, but then all hosts are denied!

Requiring Authorization

Besides restricting access by IP address, directories can be password-protected by requiring a valid user name and password combination. If a directory is protected, users need authorization to access any files in that directory. Access can only be restricted on a per directory (not per file) basis.

The following directives apply to user authorization:

- AuthType
- AuthName
- AuthUserFile
- AuthGroupFile
- Require (in a Limit section)

In order for password protection to be enforced, an access configuration file must contain all of the above directives except AuthGroupFile, which is only necessary when allowing access to a group of individuals.

The AuthType directive specifies the kind of authorization in effect for the directory. Currently, only the Basic type is implemented.

AuthType

- Usage: AuthType Basic
- Default: none

The AuthName directive specifies the reason for requiring access. The name is displayed on the user's browser when the user is asked for a user name and password, and gives some indication as to why authorization is necessary.. Of course, if you want to keep unauthorized users totally in the dark, you can specify AuthName Secret or something equally unclear to potential users. The specified name may contain spaces, as in the example.

AuthName

- Usage: AuthName *name*
- Example: AuthName Policy Working Group
- Default: none

In order to enforce password security, the Web server needs a list of acceptable user names and passwords. These are contained in a password file, or authorized user file, specified by the AuthUserFile directive.

AuthUserFile

- Usage: AuthUserFile file
- Example: AuthUserFile /webaccess/working_group/passwd
- Default: None

The AuthUserFile specifies a physical path to a user authorization file containing user names and passwords of authorized users.

User authorization files are created with the htpasswd utility included in the Web server support directory. To create a new password file, type **htpasswd -c file username**, where *file* is the name of the file to create and *user* is the password. To add a user to an existing file or to change a user's password, simply omit the -c option.

> Passwords created with `htpasswd are totally unrelated to network login and system passwords. They are stored in encrypted format on the server;`

someone reading the AuthUserFile cannot obtain passwords. However, when passwords are transmitted from client to server they are only UUEncoded. Users can employ programs called *sniffers* to detect certain types of traffic on the Internet. It is unlikely, but not impossible, for someone to capture your password by sniffing password traffic and UUDecoding it. Consequently, you should choose a password for Web authorization that is unrelated to any other password you may have.

In addition to requiring a single user name and password, standard servers can require membership in a group of authorized users. To implement group security, you must create a user authorization file as usual containing names and passwords. This can be a new file or an existing one. In addition, you must create a group authorization file is defined by the AuthGroupFile directive.

AuthGroupFile

- Usage: AuthGroupFile *file*
- Example: AuthGroupFile c:\webauth\working_group.pwd
- Default: none

The AuthGroupFile directive defines groups of users whose user names and passwords are stored in a user authorization file. Group files are plain text files that have the following format on each line:

- Group:user1 user2 user3 user4 [rl]

You can define multiple groups in one group file.

The previous directives have merely set up the locations of files to be used in password protection. The directive that actually enforces this protection is the Require directive, which occurs in a Limit section and requires that all accesses to files the restricted directory may be accompanied by a valid user name and password.

Require (in a Limit section)

- Usage: Require {user|group} *name*

The Require directive pulls together all the other authorization directives to enforce user authorization. If an individual is specified, Require looks for that user's information in the AuthUserFile specified for the directory. If membership in a group is required, Require checks to see that the user is in the specified group as listed in AuthGroupFile.

Examples:

```
<Directory c:\httpd\htdocs\secrets>
AuthType Basic
AuthName Secrets
AuthUserFile c:\auth\user\user.auth
AuthGroupFile c:\auth\user\group.auth
<LimitGET>
Require user Dickens
</Limit>
</Directory>
```

In this example, any user attempting to access a file in the secrets directory is asked to provide authorization for Secrets. If the user enters user name Dickens and the correct password in AuthUserFile, access is granted.

```
<Directory c:\httpd\htdocs\more_secrets>
AuthType Basic
AuthName More Secrets
AuthUserFile c:\auth\user\user.auth
AuthGroupFile c:\auth\user\group.auth
<Limit GET>
Order deny, allow
Deny from all
Allow from .hp.com
Require group Wise
</Limit>
</Directory>
```

In this example, a user attempting to access a file in the more_secrets directory would be asked to provide authorization for More Secrets. In order for access to be granted, the user's user name must appear in group Wise in AuthGroupFile and the user name and password must be valid in AuthUserFile. In addition, the user's IP address must end in "hp.com" or else access is denied.

Local Directory ACFs

Local directory ACFs allow most Web server features to be applied to individual directory trees. This includes features pertaining to MIME types, directory indexing, and security. The name of the Access Configuration File in each directory is given by the AccessFileName directive in the server resource map.

Local ACFs can use all the same directives as the global file with the exception of Directory and AllowOverride.

Security Features

Local ACFs provide a more convenient way to edit security and document features on a per directory basis than editing the global access configuration file. In addition, through the use of the AllowOverride directive, the administrator who maintains the global ACF can grant permission in the global ACF to use some or all of the features of local ACFs. This way, you can preserve essential security, but control less dangerous features such as indexing in each directory.

> Directives appearing in access configuration files, whether the global ACF or per directory ACFs, apply recursively to all subdirectories.

Document and Indexing Features

You can specify many of the directives in the server resource map in the global and local directory ACFs. This allows you to modify various document and indexing options for each directory. The server resource map directives that you can include in ACFs are listed below:

- DefaultType
- AddType
- AddEncoding
- ReadmeName
- DefaultIcon
- AddIcon
- AddDescription (Directory ACFs only)

- IndexIgnore

Disabling Local Directory ACFs

The administrator of the global ACF can selectively disable features in the local directory ACFs using the AllowOverride directive in each directory section. This directive can only appear in the global ACF.

AllowOverride

- Usage: AllowOverride *options*
- Example: AllowOverride FileInfo options
- Default: All

The AllowOverride directive allows local directory ACFs to override some or all of the settings in the global ACF. You can specify any of the following options:

- None
- All
- Options (indexes, includes, and so on)
- FileInfo (AddType, AddEncoding)
- AuthConfig (AuthType, AuthName, AuthUserFile, AuthGroupFile)
- Limit (Order, Deny, Allow, Require)

In order to give authors of documents on your server the capability to password protect their document directories, you must allow both the AuthConfig and Limit directives in local ACFs. Be aware, however, that this also gives document authors the capability to use the access control provisions of the Limit directive to restrict or unrestrict access from Internet addresses.

Configuring Windows NT

The Windows NT Web servers depend on the configuration of the underlying operating system to varying degrees. This section examines all of the necessary steps for configuring Windows NT to maximize the performance and security of your site.

This chapter assumes that you have already successfully installed Windows NT on your system and that you have a NIC (Network Interface Card) installed.

You can stay up-to-date on Windows NT Server through the Windows NT Server home page at **http://www.microsoft.com/NTServer/**. The Windows Workstation home page is located at: **http://www.microsoft.com/NTWorkstation/**

Windows NT Versions

There are two versions of Windows NT: NT Server and NT Workstation. The major differences between the two pertain to networking which is an important issue for you to consider. With the standard NT operating system you get peer-to-peer networking, server networking, remote access services and all the administration tools typically needed to maintain and manage a network server.

Both versions are built around the same Core NT kernel and both feature C2-level security. NT workstation is meant to be utilized as an individual computing environment or as a small workgroup server. NT Server is meant to be used as an enterprise-wide server platform. The NT server feature set includes a greater degree of fault tolerance and greater network capabilities. Remote administration is particularly salient benefit for the busy Webmaster coordinating among remote sites.

> Although NT Server and NT Workstation use the same basic kernel, NT Server is optimized to be a file and print server, and NT Workstation is optimized as a single user operating system.

If you are a serious Web publisher, you should run NT server rather than NT workstation. Also note that some Web servers including the Microsoft IIS server will not run on NT workstation, only NT Server.

> Microsoft publishes upgrades and enhancements to the NT operating systems through distributions known as Service Paks. Be sure and stay current with service pack distributions as they often solve numerous common problems. Many of the Web servers for NT will not operate without the latest service paks installed.

> There has been much confusion over whether users of Internet services were considered clients and therefore needed licenses. Recently Microsoft has clarified its position and stated that, essentially, users connecting to your Web server do not need to be licensed.

Basic Concepts

The essentials of setting up and maintaining Windows NT for your Web server revolve around the following concepts:

- Network Setup
- Security
- Administration
- Trouble Shooting

Windows NT Network Setup

The Internet is a worldwide collection of individual Transmission Control Protocol/Internet Protocol (TCP/IP) networks. Each computer on the Internet has a unique address (IP address). Information is transmitted on the Internet in data packets. Each packet is addressed to a specific computer's IP address, such as 10.212.57.189.

Because IP addresses are difficult to use and remember, the Domain Name System (DNS) was created to pair a specific IP address, such as 205.182.161.5, with a friendly domain name, such as www.Criticalmass.com (shameless plug for my own Web server!). When a user browses the Internet by using a domain name, the browser first must contact a DNS server to resolve the domain name to an IP address, and then contact the computer with that address.

This has three implications for your Web Server:

- You must install TCP/IP on your server.
- You must have a permanent IP address assigned to your server on the Internet.
- You should register a domain name in the DNS for your permanent IP address. Without a domain name, your users will need to access your site by remembering your IP Address. Ugh!

Your ISP will generally provide your IP addresses and may also register your domain names. Contact the Internet Network Information Center (InterNIC) at **http://rs.internic.net** or your ISP for more information about DNS registration.

Setting up TCP/IP

See "Building the Perfect Server,"(Ch.3) for information regarding network adapter recommendations.

Because the language of the Internet is TCP/IP, it is necessary for you to install and bind the TCP/IP protocol to your network adapter.

Like most other networking software for windows NT, TCP/IP is installed through the network applet in the control panel. Before beginning installation you will need to gather some information:

- The IP address(es) to be assigned to your network card
- The IP address of your default gateway
- The IP address of the DNS servers you will use
- Your Domain name

If you are building an intranet based system you may also need:

- Your primary and secondary WINS server IP addresses (if any)
- The LMHOST file for your network (if any)

To install TCP/IP:

1. Open the network applet in the control panel and click on the add software button. See figure 5.2.

Fig. 5.2 - Windows NT Network Settings dialog box.

1. Select TCP/IP Protocol and Related Components and the dialog box in figure 5.3 will appear. Select any additional components you desire. Click on Continue and you will be prompted for the location of the files (most likely the CD-ROM distribution disk if your NT software).

/Fig. 5.3 - Windows NT TCP/IP Installation dialog box.

1. Upon exiting the network configuration applet you will be prompted by the SNMP configuration dialog box. Unless you are familiar with SNMP and understand the settings that you need, simply select the OK button.
2. You will be prompted to setup any additional services (such as FTP) that you selected in the

TCP/IP dialog box.

If you are installing the Microsoft IIS server, do *not* install the FTP service that comes with Windows NT Server. They will conflict and neither FTP service will work.

1. After you have configured the TCP/IP services you will be shown the TCP/IP dialog box shown in figure 5.4.

/Fig. 5.4 - IP Address Configuration dialog box. Make sure these numbers are correct!

1. If you're using DHCP to assign an IP address (this is highly unlikely unless you are on a corporate NT based LAN or WAN), check the Enable Automatic DHCP configuration check box. Otherwise, Enter the TCP/IP address for the network card shown in the adapter field. If you don't know the TCP/IP address, contact your Internet Service Provider or corporate network manager. Your TCP/IP *address must be unique*. Do not enter in a random TCP/IP address, it will cause unpredictable behavior.

If you intend to host multiple Webs on this host that are accessed via separate domain names, you will need to attach multiple IP addresses to your network adapter(s). This is partly how the concept of "virtual Web servers" is implemented. Each IP address will be matched up with a domain name and during Webserver configuration we will specify which files are to be "served" for each domain.

You can attach up to 256 IP addresses per network adapter, but the TCP/IP dialog box will only configure 5 per adapter. To go beyond this requires you to edit the Windows NT registry directly. This is recommended only if you have experience editing the registry! Improper changes to the registry can cause your system to fail.

Adding More Than Five IP Addresses to a Network Card

Now that we've properly warned you, here's how to add more than five IP addresses to a network card:

You must use Regedt32.exe to assign more than five IP addresses to a network adapter card. You need to know the network adapter card name and card number used in the Registry. You can get the name from the manufacturer of your network adapter card. For example, the first Intel EtherExpress PRO card installed in a computer running Windows NT uses the name EPRO1.

To configure more that five IP addresses on a network card:

1. Start Regedt32.exe and open:
2. HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<net card name><card

number>
3. \Parameters\Tcpip
4. Double-click the IPAddress value.
5. Type every IP address you want assigned to the network adapter card. Each address must separated by a space or carriage return.
6. If applicable, double-click the SubnetMask value.
7. Type subnet masks for the IP addresses added in step 3. Pair the subnet masks to an IP address by entering the subnet masks in the same order as the IP addresses.

Many users have reported that using more than 15 or 16 IP addresses per network adapter has resulted in unstable system behavior. Others have reported using hundreds with no ill side effects. Be warned, your mileage may vary!

1. Enter your subnet mask into the appropriate field. You will get the mask from the same entity that provided your TCP/IP address. The most common scenario is that you will be using a class C address. The subnet mask will then be 255.255.255.0
2. In the default gateway field enter in the TCP/IP address of the router or computer that connects your Web host to the Internet.
3. If you are using WINS to assign NetBIOS names(again, unlikely if you are not on a corporate LAN / WAN) enter the IP addresses of the WINS servers as appropriate.
4. 10. If you have access to a Domain Name Server (if you are on the Internet you will need this) Click on the DNS button to configure the DNS addresses. See figure 5.5.

You will want to use the DNS servers that are as few network hops away as possible. Don't pick a DNS server that a friend happens to be using in Duluth. Ask your ISP or your network administrator for the closest DNS server. Your computer may communicate with the DNS server frequently, so you are wise to minimize the amount of time this takes.

Fig. 5.5 - Domain Name Service Configuration.

1. 11. Enter a name for your system in the Host Name field. If you have not been assigned one, simply put in a descriptive name such as "Webhost".
2. 12. In the domain name field enter in the name of the DNS domain that this host will be part of. Be careful not to confuse this with an NT server domain, which is entirely different.
3. 13. In the Domain Name Service (DNS) Search Order box, enter the IP addresses of the domain servers on your network. Place the DNS server closest to this station on the network (or Internet) to speed up name resolution.
4. 14. Click OK and reboot the NT operating system for these changes to take effect.

Securely Configuring Windows NT Server

The Internet is often depicted as a cyber-frontier, wilder and woolier than even the wild, wild west of yesteryear. This may be somewhat overblown by slanted media coverage, but it is a serious topic for all Webmasters. Windows NT provides user-account security and Windows NT File System (NTFS) file-system security. You can use the topics below as a checklist to ensure you have effectively used

User Accounts and NTFS to secure Windows NT Server. Additionally, you can prevent security breaches by properly configuring the services running on your computer.

Preventing Intrusion by Setting Up User Accounts

Windows NT security helps you protect your computer and its resources by requiring assigned user accounts. You can control access to all computer resources by limiting the user rights of these accounts.

Every operation on a computer running Windows NT identifies who is doing the operation. For example, the user name and password that you use to log on to Windows NT identifies who you are and defines what you are authorized to do on that computer.

What a user is authorized to do on a computer is configured in User Manager by setting User Rights in the Policies menu. User rights authorize a user to perform certain actions on the system. Your Web service should be logged on "locally" utilizing a user name that is restricted to access only what is necessary to provide services. Create an account (I call mine WEBLOGON) specifically for this purpose.

Choose Difficult Passwords

The easiest way for someone to gain unauthorized access to your system is with a stolen or easily guessed password. Make sure that all passwords used on the system, especially those with administrative rights, have difficult-to-guess passwords. In particular make sure to select a good administrator password (a long, mixed-case, alphanumeric password is best) and set the appropriate account policies. Passwords can be set by using the User Manager utility, or at the system logon prompt.

Maintain Strict Account Policies

The User Manager utility provides a way for the system administrator to specify how quickly account passwords expire (which forces users to regularly change passwords), and other policies such as how many bad logon attempts will be tolerated before locking a user out. Use these policies to manage your accounts, particularly those with administrative access, to prevent exhaustive or random password attacks.

Limit the Membership of the Administrator Group

By limiting the members of the Administrator group, you limit the number of users who might choose bad passwords and expose your system.

NTFS File Security

In addition to user accounts, you should place your data files on an NTFS partition. NTFS provides security and access control for your data files. You can limit access to portions of your file system for specific users and services by using NTFS. In particular, it's a good idea to apply Access Control Lists (ACLs) to your data files for any Internet publishing service.

The NTFS file system gives you very granular control on files by specifying users and groups that are permitted access and what type of access they may have for specific files and directories. For example,

some users may have Read-only access, while others may have Read, Change, and Write access. You should ensure that the WEBLogon account or authenticated accounts are granted or denied appropriate access to specific resources.

> You should note that the group Everyone contains all users and groups. By default, the group Everyone has full control of all files created on an NTFS drive. You will want to change this to suit your needs.

You should review the security settings for content and CGI directories and adjust them appropriately. Generally you should use the settings in the following table:

Directory Type	Suggested Access
content	Read access
programs	Read and Execute access
databases	Read and Write access

Enable Auditing

You can enable auditing of NTFS files and directories on Windows NT Server through the File Manager. You can review the audit records periodically to ensure that no one has gained unauthorized access to sensitive files.

Running Other Network Services

You should review all of the network services that you are using on any computer connected to the Internet. When reviewing consider two things:

- Is this service necessary?
- Does it create any security holes?
- Is it running under an account name with appropriate security?

> *Run only the services that you need*! The fewer services you are running on your system, the less likely a mistake will be made in administration that could be exploited. Use the Services applet in the Windows NT Control Panel to disable any services not absolutely necessary on your Internet server.

> Unbind unnecessary services from your Internet adapter cards to improve performance and security.
>
> Use the Bindings feature in the Network applet in the Windows NT Control Panel to unbind any unnecessary services from any network adapter cards connected to the Internet. For

example, you might use the Server service to copy new images and documents from computers in your internal network, but you might not want remote users to have direct access to the Server service from the Internet. If you need to use the Server service on your private network, the Server service binding to any network adapter cards connected to the Internet should be disabled. You can use the Windows NT Server service over the Internet; however, you should fully understand the security implications and licensing issues.

The FTP Server service included with Windows NT should also be disabled (this is required if the Microsoft Internet Information Server FTP service will be installed) or configured to ensure adequate security.

Check Permissions on Network Shares

If you *are* running the Server service on your Internet adapter cards, be sure to double-check the permissions set on the shares you have created on the system. It is also wise to double-check the permissions set on the files contained in the shares' directories to ensure that you have set them correctly.

NT Management Tools

NT ships with two management tools that will come in extremely handy during the day to day management of your Web server. These are the Event Log and the performance monitor. Take the time to understand the information these services can report back to you and you will go a long way towards gaining complete control of your system.

Tracking Problems with Event Viewer

Event Viewer, in the Administrative Tools program group can notify administrators of critical events by displaying pop-up messages, or by simply adding event information to log files. The information allows you to better understand the sequence and types of events that led up to a particular state or situation. You can use Event Viewer to view and manage three separate types of logs:

- System
- Security
- Application

The System log reports information regarding system level issues events. These include system startup and shutdown. See figure 5.6.

Fig. 5.6 - The System Log.

The Security log can be used to monitor all logons to your server. This is a critical log for you to monitor if your server is on the Internet. See figure 5.7.

Fig. 5.7 - The Security Log.

The Application log is used by programs running on the server to report information. Your Web server will probably report startups and shutdowns, as well as statistics and errors. See figure 5.8.

Fig. 5.8 - The Application Log.

Monitoring Your Server with Performance Monitor

The Performance Monitor, also found in the Administrative Tools program group provides a way of measuring and monitoring system performance. Many NT based servers and services automatically install Windows NT Performance Monitor counters. With these counters you can use the Windows NT Performance Monitor for real-time measurement of your Internet service use. See figure 5.9.

Fig. 5.9 - Performance Monitor for a busy server!

An excellent example of this is the Web service included in the Microsoft IIS package. This server provides counters to monitor a vast array of performance elements. These include:

- Bytes Sent/sec
- Bytes Total/sec
- CGI Requests
- Connection Attempts
- Connections/sec
- Current Anonymous Users
- Current ISAPI Requests
- Current CGI Requests
- Current Connections
- Current NonAnonymous Users
- Files Received
- Files Sent
- Files Total
- Get Requests
- Head Requests
- Logon Attempts
- Maximum Anonymous Users
- Maximum ISAPI Requests
- Maximum CGI Requests
- Maximum Connections
- Maximum Non Anonymous Users
- Not Found Errors
- Other Request Methods
- Post Requests
- Total Anonymous Users
- Total Non Anonymous Users

While running your server, you should get to know the performance monitor and establish benchmarks for standard behavior. When these performance benchmarks are deviated from, you can decide whether action is warranted. Key areas to monitor are CPU usage and disk requests.

Optimizing Windows NT Performance

One of the advantages of Windows NT is that much of the work to optimize is done for you automatically by the system. A few of the most common methods for boosting performance are covered briefly below.

- *RAM* Most NT GURU's will tell you that the easiest way to boost system performance is to add RAM. You should have a minimum of 16 MB to run Windows NT and 32 is recommended. If you are running a highly active server that uses CGI programming you will likely see nice performance increases with each addition of RAM. Many of the popular NT Web sites use 128 MB of RAM, and sometimes more.
- *Hard Disk* A Web site lives and dies by its hard disk. The first rule in hard disks today is to get a hard drive with a fast access rating. Drive speeds are usually rated in milliseconds and in today's

environment a 5 to 9 millisecond rating is pretty good. The second rule in hard drives is to connect it to your system through a 32-bit interface. Your server should have a 32-bit disk controller either built in or available via a 32-bit slot. The third rule is to defragment your hard drive. This can greatly speed up file access by placing your files in contiguous blocks on the hard drive. Unfortunately there are only a few defragmenters on the market for Windows NT, and all of them are expensive.

Many experts will tell you that increasing the performance of NT is simply a function of adding RAM. My experience has shown me that RAM is definitely your best value when speeding up your system.

- *CPU* Serving Web is generally not much of a strain on the microprocessor. The CPU is only likely to become a factor as you start running sophisticated CGI type applications. If you plan on a busy Web site, utilize a fast Pentium and consider the Pentium Pro.
- *Virtual Memory* Windows NT utilizes a virtual memory system. This means that when necessary, NT utilizes hard disk space as RAM allowing more and larger applications to run. NT uses a paging file on each logical drive for virtual memory. The size and location of these files in configurable through the system applet on the control panel. Be sure that you have plenty of available hard disk space available. If you notice that your NT system is constantly accessing the hard drive during normal activity, this is an indication that you need more RAM.

Run only 32-bit applications. Windows NT is a 32-bit operating system, that is backwards compatible to allow you to run 16-bit applications. 16-bit applications cause NT to run a sub-system for compatibility. This will slow down all of the applications that are currently executing on your system.

Configuring Windows 95

Configuring Windows 95 to run a Web server is relatively simple when compared to Windows NT. The fact is that as a server platform Windows 95 is adequate for only smaller and simpler Web sites. That doesn't mean that you can't create a web site that's perfect for your individual needs, just scale your expectations accordingly.

This chapter assumes that Windows 95 has been successfully installed on your PC. This chapter also assumes that you have a NIC (Network Interface Card) installed on a LAN connected to the Internet.

It is also feasible to connect your server to the Internet via dial-up modem. This is done in many low volume cases due to the low cost relative to a permanent circuit Most of the steps here are similar to those required for a modem connection. If you have a modem installed you will see it referred to as a Dial-up Adapter in the network configuration (see figure 5.10). Simply substitute *Dial-up Adapter* where you see *NIC*. We recommend that you eliminate most or all graphics from your site if serving via dial-up.

We will configure Windows 95 to run the WebQuest server included on the Webmaster CD.

The configuration steps this section covers are:

- Network Setup
- Security

These configuration needs are the same as those covered previously in the section on Windows NT. In fact we will often refer you there for the background material already covered.

Network Setup

The Internet uses the TCP/IP network protocols to establish and maintain communications between computers. You can review our discussion of this earlier in this chapter. See "Windows NT Network Setup,"(Ch.5)

Installing TCP/IP

To install TCP/IP:

1. Open the network applet in the Windows 95 control panel and click on the Add Software button.

Fig. 5.10 - Windows 95 Network Configuration Applet.

1. Single click to highlight your NIC (Network Adapter Card) and then click on the Add button. See figure 5.11.

Fig. 5.11 - Windows 95 Network Component Dialog Box.

1. Select Protocol and click the Add button.

This will bring up the dialog box in figure 5.12. This is a list of network protocols that come with Windows 95, ordered by manufacturer. In most cases you will want to use the native Microsoft TCP/IP component. In a corporate environment you will need to verify this with your network administrator.

Fig. 5.12 - Windows 95 Network Protocol dialog box.

1. Select Microsoft in the left-hand list box as shown in figure 5.12. Click on TCP/IP in the Network Protocols box on the right. Click the OK button.
2. Windows will now automatically bind the TCP/IP network protocol to your NIC. You will see a new entry in the Network Configuration dialog that begins with TCP/IP -> as shown in figure 5.11.
3. Highlight the new TCP/IP binding and click on the Properties button. This will bring up the TCP/IP properties dialog box.
4. The initial configuration page is the TCP/IP address page.

Unlike Windows NT you can only bind one TCP/IP address per Network Interface card.

This makes Windows 95 a poor choice to serve Virtual Webs.

1. Enter the TCP/IP Address assigned to this PC and the appropriate Subnet Mask. If you don't know this information contact your ISP or network administrator.
2. Select the Tab titled Gateway as shown in figure 5.13. Enter the TCP/IP address of the remote gateway. This information should be obtained from your network administrator or ISP.

Fig 5.13 - TCP/IP Gateway dialog box.

1. Select the Tab titled DNS Configuration as shown in figure 5.14 and enter in the DNS Server IP addresses as given to you by your ISP or network administrator.

Fig. 5.14 - DNS configuration dialog box.

1. In the host and domain boxes of the DNS Configuration dialog enter the name of your PC and the TCP/IP domain you belong to. If you don't have a name assigned to you from your network administrator simply enter any name you wish. This name is public and will be received by remote machines under certain circumstances so be prudent.
2. 10. Click the OK button and return to the Network Configuration dialog. You have finished configuring the physical network access for your web server. Select the OK button and reboot your PC. You are now ready to install your WEB Server software.

Security

See " Access Controls on a Windows 95 System,"(Ch.8)

Setting up network security for your Web site is accomplished from the same Network Configuration applet that you used for the physical network configuration. This is covered in detail in Chapter 8.

configure the destination web server to listen for requests for HTTPS sessions on port 80 and to direct the affected clients' browsers to request information using a resource locator of the form "https://www.domain.com:80".

The technical staff of every e-commerce web site that attempts to do business with the general public ought eventually to encounter the Firewall Problem since some customers and some potential customers spend some time at offices or other locations where their computers are connected to the Internet through firewalls that block outgoing packets addressed to destination port 443.

Consequently, if Applicant's invention should be obvious to anyone skilled in the art who encounters the Firewall Problem, then it would be logical to expect that, during the more than six years since Applicant first reduced Applicant's invention to practice and the more than three years since Kalajan issued, the technical staffs of many e-commerce web sites that seek to do business with the general public would either (i) have duplicated Applicant's invention or (ii) have settled upon some other solution to the Firewall Problem that permits secure communication with affected customers' computers.

However, Applicant is not personally aware of any web sites that direct a customer's browser to a resource locator of the form https://www.domain.com:80 or implement some other solution to the Firewall Problem that permits secure communication with customers' computers that are affected by the Firewall Problem.

Consider for example the web sites barnesandnobel.com and amazon.com – two popular, highly competitive, technologically savvy e-commerce web sites that seek to conduct business with the general public.

Based on tests conducted by Applicant on December 29, 2004, it appears that when confronted with the Firewall Problem, the persons skilled in the art employed by

barnesandnobel.com decided to drop back and punt. To ensure security, barnesandnobel.com uses SSL (i.e., HTTPS) for order submission and the collection of credit card information. If the computer used by a potential customer of barnesandnobel.com is connected to the Internet through a firewall that blocks outgoing packets addressed to destination port 443, then the potential customer is allowed to fill up a shopping cart but at checkout time the potential customer's browser will display an unhelpful error message as soon as the customer's browser is directed to establish an HTTPS session using the default destination port of 443.

Based on tests conducted by Applicant on December 28, 2004, it is clear that when confronted with the Firewall Problem, the persons skilled in the art employed by amazon.com have also failed to duplicate Applicant's invention or to implement some different solution that permits encrypted communication with affected customers. The folks at amazon.com clearly recognize the Firewall Problem, warn customers about it, and offer affected customers the choice of giving up or submitting order and payment details in an unsecured manner (i.e., using HTTP rather than HTTPS).

In particular, the last page of the check out process that amazon.com normally sends to customers without encryption (i.e., using HTTP) contains both a button labeled:

"Sign in using our secure server"

and a link that says

"The secure server will encrypt your information. If you received an error message when you tried to use our secure server, sign in using our standard server."

A copy of that page as viewed by the Applicant on December 28,

2004 is being filed with the Supplemental IDS filed on January 11, 2005. The above described button is a shaded oval near the middle of the web page (about 1/3 of the way down the printed page). The above described "standard server" link is near the bottom of the web page (a bit less than half-way down the printed page).

If a customer should click on the button labeled "Sign in using our secure server", then such customer's browser would be directed to a URL of the form "https://www.amazon.com/*", which by default implies destination port 443. If such customer's computer should be connected to the Internet through a firewall that blocks outgoing packets addressed to destination port 443 and such customer clicks on such button, then such customer would see an uninformative error message.

If a customer should click on the "standard server" link, then such customer's browser would be directed to a URL of the form "http://www.amazon.com/*" which by default implies destination port 80, thereby avoiding part of the Firewall Problem. Unfortunately, since that URL begins with "http", the remainder of the checkout process, including the transmission of credit card information, would then be conducted using HTTP which is NOT encrypted for security.

Since popular e-commerce sites that seek to do business with the general public neither (i) routinely use URLs of the form "https://www.securedomain.com/*:80" for the secure portions of their check out procedures nor (ii) routinely use some other solution to the Firewall Problem that ensures secure, encrypted communications with affected customers' computers, Applicant contends that Applicant's invention was not obvious when it was first reduced to practice by Applicant and remains non-obvious today, more than six years later.

(1-c-4) The Firewall Problem and Applicant's invention are in the field of making it easier for a client to access a server, but Kalajan is in the field of making it harder for clients to access a server.

amazon.com.

SIGN IN SHIPPING & PAYMENT GIFT WRAP PLACE ORDER

Ordering from Amazon.com is quick and easy

Enter your e-mail address: amazon@nbw.com

○ **I am a new customer.**
(You'll create a password later)

◉ **I am a returning customer, and my password is:**

Sign in using our secure server

Forgot your password? Click here

Has your e-mail address changed since your last order?

The secure server will encrypt your information. If you received an error message when you tried to use our secure server, sign in using our standard server.

The only way to place an order at Amazon.com is via our Web site. (Sorry--no phone orders. However, if you prefer, you may phone in your credit card number *after* filling out the order form online.)
Redeeming a gift certificate? We'll ask for your claim code when it's time to pay.
Having difficulties? Please visit our Help pages to learn more about placing an order.
